```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
```

```
XX      XX  FFFFFFFFFF  DDDDDDDD    RRRRRRRR     IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  FFFFFFFFFF  DDDDDDDD    RRRRRRRR     IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  FF          DD      DD  RR      RR     II      VV      VV  EE          RR      RR
XX      XX  FF          DD      DD  RR      RR     II      VV      VV  EE          RR      RR
  XX  XX    FF          DD      DD  RR      RR     II      VV      VV  EE          RR      RR
  XX  XX    FF          DD      DD  RR      RR     II      VV      VV  EE          RR      RR
    XX      FFFFFFFF    DD      DD  RRRRRRRR       II      VV      VV  EEEEEEEE    RRRRRRRR
    XX      FFFFFFFF    DD      DD  RRRRRRRR       II      VV      VV  EEEEEEEE    RRRRRRRR
  XX  XX    FF          DD      DD  RR  RR         II      VV      VV  EE          RR  RR
  XX  XX    FF          DD      DD  RR  RR         II      VV      VV  EE          RR  RR
XX      XX  FF          DD      DD  RR    RR       II        VV  VV    EE          RR    RR        ....
XX      XX  FF          DD      DD  RR    RR       II        VV  VV    EE          RR    RR        ....
XX      XX  FF          DDDDDDDD    RR      RR     IIIIII      VV      EEEEEEEEEE  RR      RR      ....
XX      XX  FF          DDDDDDDD    RR      RR     IIIIII      VV      EEEEEEEEEE  RR      RR      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII    SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  XFDRIVER - DR32 DRIVER
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY:     EXECUTIVE, I/O DRIVERS
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;       THIS MODULE IS THE DRIVER FOR THE DR32.
0000    34  ;
0000    35  ; ENVIRONMENT:  KERNEL MODE, NON-PAGED
0000    36  ;
0000    37  ; AUTHOR:  STEVE BECKHARDT,     CREATION DATE:  23-FEB-1979
0000    38  ;
0000    39  ; MODIFIED BY:
0000    40  ;
0000    41  ;       V03-002 TCM0002         Trudy C. Matthews       29-Feb-1983
0000    42  ;               Update CPUDISP that determines if this is a DR780 or a
0000    43  ;               DR750 to work on an 11/790 (take the DR780 path).
0000    44  ;
0000    45  ;       V03-001 EAD0068         Elliott A. Drayton      01-Jul-1982
0000    46  ;               Add code to prevent race condition with HALT bit.
0000    47  ;
0000    48  ;       V02-012 TCM0001         Trudy C. Matthews       31-Jul-1981
0000    49  ;               Change all "7ZZ"s to "730"s.
0000    50  ;
0000    51  ;       V02-011 EAD0011         Elliott A. Drayton      13-Feb-1981
0000    52  ;               Add code to raise and lower IPL to prevent
0000    53  ;               race condition.
0000    54  ;
0000    55  ;       V02-010 SRB0006         Steve Beckhardt         17-Sep-1979
0000    56  ;               Modified the driver to support the DR750.
0000    57  ;
```

```
0000     58 ;--
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;
                        $ACBDEF                 ; AST CONTROL BLOCK OFFSETS
                        $ADPDEF                 ; ADP OFFSETS
                        $CRBDEF                 ; CRB OFFSETS
                        $DCDEF                  ; DEVICE CLASSES AND TYPES
                        $DDBDEF                 ; DDB OFFSETS
                        $DPTDEF                 ; DRIVER PROLOGUE TABLE DEFINITIONS
                        $DYNDEF                 ; STRUCTURE TYPE CODE DEFINITIONS
                        $EMBDEF                 ; EMB OFFSETS
                        $FKBDEF                 ; FKB OFFSETS
                        $IDBDEF                 ; IDB OFFSETS
                        $IPLDEF                 ; IPL DEFINITIONS
                        $IODEF                  ; I/O FUNCTION CODES
                        $IRPDEF                 ; IRP OFFSETS
                        $IRPEDEF                ; IRPE OFFSETS
                        $PCBDEF                 ; PCB OFFSETS
                        $PRDEF                  ; PROCESSOR REGISTER DEFINITIONS
                        $PRIDEF                 ; PRIORITY INCREMENT CLASS DEFINITIONS
                        $PRTDEF                 ; PROTECTION CODE DEFINITIONS
                        $PTEDEF                 ; PTE DEFINITIONS
                        $SSDEF                  ; SYSTEM STATUS VALUES
                        $UCBDEF                 ; UCB OFFSETS
                        $VADEF                  ; VIRTUAL ADDRESS FIELD DEFINTIONS
                        $VECDEF                 ; INTERRUPT DISPATCH VECTOR OFFSETS
                        $XFDEF                  ; DR32 DEFINITIONS

                ;
                ; MACROS:
                ;

                ;
                ; EQUATED SYMBOLS:
                ;

                ;
                ; QIO ARGUMENT LIST OFFSETS
                ;

00000000        P1=0                            ; 1ST FUNCTION DEPENDENT PARAMETER
00000004        P2=4                            ; 2ND      "         "         "
00000008        P3=8                            ; 3RD      "         "         "

                ;
                ; MISC. DEFINITIONS
                ;

000000FB        DR780_MAXRATE=251               ; MAXIMUM DATA RATE VALUE FOR DR780
000000FC        DR750_MAXRATE=252               ; MAXIMUM DATA RATE VALUE FOR DR750

00000040        IRP$L_PKTASTADR=IRP$L_ABCNT     ; OFFSET OF PACKET AST ADDRESS IN IRP
00000044        IRP$L_PKTASTPRM=IRP$L_OBCNT     ; OFFSET OF PACKET AST PARAMETER
0000003C        IRP$B_RATE=IRP$L_IOST2          ; OFFSET OF DATA RATE
0000003D        IRP$B_FLAGS=IRP$L_IOST2+1       ; OFFSET OF COMMAND TABLE FLAGS
```

```
00000044  0000   117 IRPE$L_CBLKADR=IRPE$L_BCNT2+4          ; OFFSET OF COMMAND BLOCK ADDRESS
00000048  0000   118 IRPE$L_BBLKADR=IRPE$L_BCNT2+8          ; OFFSET OF BUFFER BLOCK ADDRESS
          0000   119
          0000   120 ;
          0000   121 ; DR32 DEVICE REGISTER DEFINITIONS
          0000   122 ;
          0000   123
          0000   124         $DEFINI DR
          0000   125 $DEF     DR_DCR          .BLKL   1       ; DR32 CONTROL REGISTER
          0004   126         _VIELD  DR_DCR,0,<-
          0004   127                 <ADPTYP,8>,-            ; ADAPTER TYPE
          0004   128                 <ID2ERR,,M>,-           ; ID2 ERROR
          0004   129                 <ID2TOC,,M>,-           ; ID2 TIMEOUT CAUSE
          0004   130                 <ID2TO,,M>,-            ; ID2 TIMEOUT
          0004   131                 <STALL,,M>,-            ; DDI STALL
          0004   132                 <ID1ERR,,M>,-           ; ID1 ERROR
          0004   133                 <ID1TOC,,M>,-           ; ID1 TIMEOUT CAUSE
          0004   134                 <ID1TO,,M>,-            ; ID1 TIMEOUT
          0004   135                 <RDS,,M>,-              ; READ DATA SUBSTITUTE
          0004   136                 <CRD,,M>,-              ; CORRECTED READ DATA
          0004   137                 <DCRHLT,,M>,-           ; DCR HALT
          0004   138                 <DCRABT,,M>,-           ; DCR ABORT INTERRUPT
          0004   139                 <PKTINT,,M>,-           ; PACKET INTERRUPT
          0004   140                 <INTENB,,M>,-           ; INTERRUPT ENABLE
          0004   141                 <,1>,-                  ; RESERVED
          0004   142                 <PWR_UP,,M>,-           ; ADAPTER POWER UP
          0004   143                 <PWR_DN,,M>,-           ; ADAPTER POWER DOWN
          0004   144                 <EXTABT,,M>,-           ; EXTERNAL ABORT
          0004   145                         -               ; NOTE:  THE NEXT 7 BITS ARE USED BY
          0004   146                         -               ; DR750 ONLY.  THE DR780 HAS THESE BITS
          0004   147                         -               ; IN THE DR_UTL REGISTER.
          0004   148                 <FCIPE,,M>,-            ; FORCE CONTROL INTERCONNECT PARITY ERR.
          0004   149                 <FDIPE,,M>,-            ; FORCE DATA INTERCONNECT PARITY ERROR
          0004   150                 <ENPEAB,,M>,-           ; ENABLE D.I. PARITY ERROR ABORT
          0004   151                 <WCSPE,,M>,-            ; WCS PARITY ERROR
          0004   152                 <CIPE,,M>,-             ; CONTROL INTERCONNECT PARITY ERROR
          0004   153                 <DIPE,,M>,-             ; DATA INTERCONNECT PARITY ERROR
          0004   154                 <PARERR,,M>,-           ; PARITY ERROR (OR OF LAST 3 BITS)
          0004   155                 >
          0004   156
          0004   157 ; DCR CONTROL FIELD A CODES (USED WHEN WRITING TO DCR)
          0004   158
00000100  0004   159 DCR_K_CLRPWRUP=^X100                    ; CLEAR POWER UP
00000200  0004   160 DCR_K_CLRPWRDN=^X200                    ; CLEAR POWER DOWN
00000400  0004   161 DCR_K_CLRABTINT=^X400                   ; CLEAR ABORT INTERRUPT
00000500  0004   162 DCR_K_CLRINTENB=^X500                   ; CLEAR INTERRUPT ENABLE
00000600  0004   163 DCR_K_SETINTENB=^X600                   ; SET INTERRUPT ENABLE
00000700  0004   164 DCR_K_CLRHLT=^X700                      ; CLEAR HALT
          0004   165
          0004   166 ; DCR CONTROL FIELD B CODES (USED WHEN WRITING TO DCR)
          0004   167
00001000  0004   168 DCR_K_CLRCRD=^X1000                     ; CLEAR CRD
00002000  0004   169 DCR_K_SETEXTABT=^X2000                  ; SET EXTERNAL ABORT
00003000  0004   170 DCR_K_CLRPKTINT=^X3000                  ; CLEAR PACKET INTERRUPT
00004000  0004   171 DCR_K_RESET=^X4000                      ; RESET
00005000  0004   172 DCR_K_SETOSQTST=^X5000                  ; SET OSEQ TEST
00006000  0004   173 DCR_K_CLROSQTST=^X6000                  ; CLEAR OSEQ TEST
```

```
                     0004   174
                     0004   175 $DEF    DR_UTL          .BLKL   1       ; UTILITY REGISTER
                     0008   176          _VIELD  DR_UTL,0,<-
                     0008   177                  <RATE,8>,-              ; DATA RATE
                     0008   178                  <,3>,-                  ; RESERVED
                     0008   179                  <VALID,,M>,-            ; WCS VALID
                     0008   180                  <,13>,-                 ; RESERVED
                     0008   181                       -                  ; NOTE:  THE NEXT 7 BITS ARE USED BY
                     0008   182                       -                  ; DR780 ONLY.  THE DR750 HAS THESE BITS
                     0008   183                       -                  ; IN THE DR_DCR REGISTER.
                     0008   184                  <FCIPE,,M>,-            ; FORCE CONTROL INTERCONNECT PARITY ERR.
                     0008   185                  <FDIPE,,M>,-            ; FORCE DATA INTERCONNECT PARITY ERROR
                     0008   186                  <ENPEAB,,M>,-           ; ENABLE D.I. PARITY ERROR ABORT
                     0008   187                  <WCSPE,,M>,-            ; WCS PARITY ERROR
                     0008   188                  <CIPE,,M>,-             ; CONTROL INTERCONNECT PARITY ERROR
                     0008   189                  <DIPE,,M>,-             ; DATA INTERCONNECT PARITY ERROR
                     0008   190                  <PARERR,,M>,-           ; PARITY ERROR (OR OF LAST 3 BITS)
                     0008   191          >
                     0008   192
                     0008   193 $DEF    DR_WCSA         .BLKL   1       ; WCS ADDRESS
                     000C   194          _VIELD  DR_WCSA,0,<-
                     000C   195                  <SEL,,M>,-              ; SELECT (LOW OR HI PART OF MICRO WORD)
                     000C   196                  <ADDR,10>,-             ; ADDRESS
                     000C   197                  <,20>,-                 ; RESERVED
                     000C   198                  <WCS,,M>,-              ; LOAD WCS FLAG (DR750 ONLY)
                     000C   199          >
                     000C   200          _VIELD  DR_WCSA,0,<-
                     000C   201                  <LSADR,11>,-            ; LOCAL STORE ADDRESS (DR750 ONLY)
                     000C   202          >
                     000C   203
                     000C   204 $DEF    DR_WCSD         .BLKL   1       ; WCS DATA
                     0010   205
00000014             0010   206                          .BLKL   1       ; RESERVED
                     0014   207
                     0014   208 $DEF    DR_SBIADR       .BLKL   1       ; SBI ADDRESS
                     0018   209
                     0018   210 $DEF    DR_SBIBCNT      .BLKL   1       ; SBI BYTE COUNT
                     001C   211
                     001C   212 $DEF    DR_DDIBCNT      .BLKL   1       ; DDI BYTE COUNT
                     0020   213
                     0020   214          ;
                     0020   215          ; DEFINE USER CONTROL REGISTER (GO BIT)
                     0020   216          ;
                     0020   217
00000200             0020   218 .=^X200                                  ; STARTS ON 2ND PAGE OF DR ADDRESS SPACE
                     0200   219
                     0200   220 $DEF    DR_USER         .BLKL   1       ; USER CONTROL REGISTER
                     0204   221          _VIELD  DR_USER,0,<-
                     0204   222                  <GO,,M>,-               ; GO BIT
                     0204   223                  <,31>,-                 ; RESERVED
                     0204   224          >
                     0204   225
                     0204   226          ;
                     0204   227          ; DEFINE LOCAL STORE ADDRESSES FOR DR780
                     0204   228          ; THESE ARE ADDRESSED AS OFFSETS FROM THE FIRST DEVICE REGISTER
                     0204   229          ;
                     0204   230
```

```
                     00000400  0204    231 .=^X400                                 ; STARTS ON 3RD PAGE OF DR ADDRESS SPACE
                               0400    232
                               0400    233 $DEF    DR_780_DSL      .BLKL   1       ; DR32 STATUS LONGWORD
                               0404    234 $DEF    DR_780_SBR      .BLKL   1       ; SYSTEM BASE REGISTER
                               0408    235 $DEF    DR_780_GBR      .BLKL   1       ; GLOBAL PAGE TABLE BASE REGISTER
                               040C    236 $DEF    DR_780_CMDBVA   .BLKL   1       ; BASE VIRTUAL ADDR. COMMAND BLOCK
                               0410    237 $DEF    DR_780_CMDLEN   .BLKL   1       ; LENGTH OF COMMAND BLOCK
                               0414    238 $DEF    DR_780_CMDSVAPT .BLKL   1       ; SVAPTE OF COMMAND BLOCK
                               0418    239 $DEF    DR_780_BFRBVA   .BLKL   1       ; BASE VIRTUAL ADDR. BUFFER BLOCK
                               041C    240 $DEF    DR_780_BFRLEN   .BLKL   1       ; LENGTH OF BUFFER BLOCK
                               0420    241 $DEF    DR_780_BFRSVAPT .BLKL   1       ; SVAPTE OF BUFFER BLOCK
                               0424    242
                               0424    243         ;
                               0424    244         ; DEFINE LOCAL STORE ADDRESSES FOR DR750
                               0424    245         ; THESE ARE ADDRESSED BY LOADING THEIR ADDRESS INTO THE DR_WCSA
                               0424    246         ; REGISTER AND READING OR WRITING THE DR_WCSD REGISTER
                               0424    247         ;
                               0424    248
                     00000000  0424    249 .=0                                     ; START OF LOCAL STORE REGISTERS
                               0000    250
                               0000    251 $DEF    DR_750_DSL      .BLKB   1       ; DR32 STATUS LONGWORD
                               0001    252 $DEF    DR_750_SBR      .BLKB   1       ; SYSTEM BASE REGISTER
                               0002    253 $DEF    DR_750_GBR      .BLKB   1       ; GLOBAL PAGE TABLE BASE REGISTER
                               0003    254 $DEF    DR_750_CMDBVA   .BLKB   1       ; BASE VIRTUAL ADDR. COMMAND BLOCK
                               0004    255 $DEF    DR_750_CMDLEN   .BLKB   1       ; LENGTH OF COMMAND BLOCK
                               0005    256 $DEF    DR_750_CMDSVAPT .BLKB   1       ; SVAPTE OF COMMAND BLOCK
                               0006    257 $DEF    DR_750_BFRBVA   .BLKB   1       ; BASE VIRTUAL ADDR. BUFFER BLOCK
                               0007    258 $DEF    DR_750_BFRLEN   .BLKB   1       ; LENGTH OF BUFFER BLOCK
                               0008    259 $DEF    DR_750_BFRSVAPT .BLKB   1       ; SVAPTE OF BUFFER BLOCK
                               0009    260
                               0009    261         $DEFEND DR
                               0000    262
                               0000    263
                               0000    264 ;
                               0000    265 ; DR32 SPECIFIC UCB OFFSETS
                               0000    266 ;
                               0000    267
                               0000    268         $DEFINI UCB
                               0000    269
                     000000A0  0000    270 .=UCB$L_DPC+4
                               00A0    271
                               00A0    272         _VIELD  UCB,0,<-                ; DEFINE BITS FOR UCB$W_DEVSTS
                               00A0    273                 <ADPPWRUP,,M>,-         ;  ADAPTER POWER UP
                               00A0    274                 <FKLOCK,,M>,-           ;  FORK INTERLOCK BIT
                               00A0    275                 <ABORT,,M>,-            ;  ABORT PENDING
                               00A0    276                 <DR750,,M>,-            ;  INDICATES DEVICE IS DR750
                               00A0    277         >
                               00A0    278
                               00A0    279 $DEF    UCB$L_DCR       .BLKL   1       ; STORED COPY OF DCR REGISTER
                               00A4    280 $DEF    UCB$L_SAVSTATUS .BLKL   1       ; SAVED STATUS FOR DRIVER ABORTS
                               00A8    281 $DEF    UCB$L_SAVDCR    .BLKL   1       ; SAVED COPY OF DCR REGISTER
                               00AC    282
                     000000AC  00AC    283 UCB$K_SIZE=.
                               00AC    284
                               00AC    285         $DEFEND UCB
                               0000    286
                               0000    287
```

```
                        ;
                        ; OWN STORAGE:
                        ;
                        ;
                        ; DRIVER PROLOGUE TABLE
                        ;
                                DPTAB   END=XF_END,-                    ; END OF DRIVER
                                        ADAPTER=DR,-                    ; ADAPTOR TYPE
                                        UCBSIZE=UCB$K_SIZE,-            ; UCB SIZE
                                        NAME=XFDRIVER                   ; DRIVER NAME
                                DPT_STORE INIT
                                DPT_STORE UCB,UCB$B_FIPL,B,8    ; FORK IPL
                        ;***
                        ; FORK IPL OK?
                        ;***
                                DPT_STORE UCB,UCB$L_DEVCHAR,L,- ; DEVICE CHARACTERISTICS
                                        <DEV$M_AVL-             ; AVAILABLE
                                        !DEV$M_ELG-             ; ERROR LOGGING ENABLED
                                        !DEV$M_RTM-             ; REAL TIME DEVICE
                                        !DEV$M_IDV-             ; INPUT DEVICE
                                        !DEV$M_ODV>             ; OUTPUT DEVICE
                                DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_REALTIME     ; DEVICE CLASS
                                DPT_STORE UCB,UCB$B_DIPL,B,22           ; DEVICE IPL
                                DPT_STORE REINIT
                                DPT_STORE DDB,DDB$L_DDT,D,XF$DDT                ; DDT ADDRESS
                                DPT_STORE CRB,CRB$L_INTD+4,D,INTERRUPT_SVC      ; INTERRUPT SERVICE
                                DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,UNIT_INIT ; UNIT INIT.
                                DPT_STORE END
                        ;
                        ; DRIVER DISPATCH TABLE
                        ;
                                DDTAB   XF,-                    ; DEVICE NAME
                                        START IO,-              ; START I/O ENTRY POINT
                                        0,-                     ; UNSOLICITED INTERRUPT
                                        FUNCTABLE,-             ; FUNCTION DECISION TABLE
                                        CANCEL_IO,-             ; CANCEL I/O
                                        REGDUMP,-               ; REGISTER DUMP ROUTINE
                                        <36+160>,-              ; SIZE OF DIAGNOSTIC BUFFER
                                        <EMB$L_DV_REGSAV+4+160> ; SIZE OF ERROR LOGGING BUFFER
                        ;
                        ; FUNCTION DECISION TABLE
                        ;
                        FUNCTABLE:
                                FUNCTAB ,<LOADMCODE,-                   ; LEGAL FUNCTIONS
                                        STARTDATA,-
                                        STARTDATAP>
                                FUNCTAB                                 ; NO BUFFERED I/O FUNCTIONS
                                FUNCTAB LOAD_MICROCODE,<LOADMCODE>      ; LOAD MICROCODE
                                FUNCTAB STARTDATA_FDT,<STARTDATA,-      ; START DATA
```

```
0054   345
0060   346

STARTDATAP>          ; START DATA PHYSICAL
```

```
                     0060   348          .SBTTL  LOAD_MICROCODE - Load microcode FDT routine
                     0060   349 ;++
                     0060   350 ; FUNCTIONAL DESCRIPTION:
                     0060   351 ;
                     0060   352 ;       This routine is an FDT routine which performs the Load Microcode
                     0060   353 ;       QIO.  It locks the microcode image in memory, verifies that the
                     0060   354 ;       device is not busy, loads and then verifies the microcode.
                     0060   355 ;       Verification consists of addressing all the locations in the WCS
                     0060   356 ;       and checking for a parity error.
                     0060   357 ;       If the microcode is loaded successfully, the the WCS valid bit
                     0060   358 ;       is set in the DR32.
                     0060   359 ;       This routine also sets the data rate to the last value stored
                     0060   360 ;       in the device dependent characteristics.
                     0060   361 ;
                     0060   362 ; CALLING SEQUENCE:
                     0060   363 ;
                     0060   364 ;       Called from the FDT routine dispatcher in the QIO system service.
                     0060   365 ;       On completion jumps to EXE$FINISHIOC.
                     0060   366 ;
                     0060   367 ; INPUT PARAMETERS:
                     0060   368 ;
                     0060   369 ;       R3      Address of I/O packet.
                     0060   370 ;       R4      Current process PCB address.
                     0060   371 ;       R5      Address of UCB.
                     0060   372 ;       R6      Address of CCB.
                     0060   373 ;       P1(AP)  Address of microcode image.
                     0060   374 ;       P2(AP)  Size (in bytes) of microcode image.
                     0060   375 ;
                     0060   376 ; IMPLICIT INPUTS:
                     0060   377 ;
                     0060   378 ;       NONE
                     0060   379 ;
                     0060   380 ; OUTPUT PARAMETERS:
                     0060   381 ;
                     0060   382 ;       R0      Contains a completion code
                     0060   383 ;
                     0060   384 ; IMPLICIT OUTPUTS:
                     0060   385 ;
                     0060   386 ;       The WCS valid bit is set in the DR32.
                     0060   387 ;
                     0060   388 ; COMPLETION CODES:
                     0060   389 ;
                     0060   390 ;       These are in addition to the ones EXE$WRITELOCK can return:
                     0060   391 ;
                     0060   392 ;       SS$_NORMAL      Successful completion
                     0060   393 ;       SS$_PARITY      Parity error detected during microcode verification
                     0060   394 ;       SS$_DEVACTIVE   Device is active
                     0060   395 ;       SS$_POWERFAIL   Device is powered down
                     0060   396 ;
                     0060   397 ; SIDE EFFECTS:
                     0060   398 ;
                     0060   399 ;       None
                     0060   400 ;
                     0060   401 ;--
                     0060   402
                     0060   403 LOAD_MICROCODE:
50   6C   D0   0060   404          MOVL    P1(AP),R0               ; Get address of microcode image
```

```
                     51    04 AC    3C   0063   405          MOVZWL  P2(AP),R1                ; Get size of image (in bytes)
                         59    50    7D   0067   406          MOVQ    R0,R9                   ; Save address and size in R9 and R10
                  00000000'GF    16   006A   407          JSB     G^EXE$WRITELOCK         ; Lock image into memory
                   64 A5    20    AA   0070   408          BICW    #UCB$M_POWER,UCB$W_STS(R5) ; Clear powerfail bit
                                        0074   409
                                        0074   410          ASSUME  IDB$L_CSR  EQ  0
                                        0074   411
                     54    24 A5    D0   0074   412          MOVL    UCB$L_CRB(R5),R4        ; Get pointer to CRB
                     54    2C B4    D0   0078   413          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get address of 1st device CSR
                                        007C   414
                                        007C   415 10$:      ; R4 contains address of 1st device CSR.  Make sure device has
                                        007C   416           ; power before accessing any device registers.  Then, if the device
                                        007C   417           ; is not busy, reset it and clear WCS valid.
                                        007C   418
                                        007C   419          ASSUME  UCB_V_ADPPWRUP  EQ  0
                                        007C   420
                   50    0364 8F    3C   007C   421          MOVZWL  #SS$_POWERFAIL,R0       ; Assume error
                        11 68 A5    E9   0081   422          BLBC    UCB$W_DEVSTS(R5),15$    ; Branch if adapter has no power
                                        0085   423
                   50    02C4 8F    3C   0085   424          MOVZWL  #SS$_DEVACTIVE,R0       ; Assume device is active
                                        008A   425          DSBINT  UCB$B_FIPL(R5)          ; Raise IPL to fork level
                03 64 A5    08    E1   0091   426          BBC     #UCB$V_BSY,UCB$W_STS(R5),17$    ; Br. if UCB is not busy
                              00A7   31   0096   427 15$:     BRW     80$                     ; Finish I/O
                   64    4000 8F    3C   0099   428 17$:     MOVZWL  #DCR_K_RESET,DR_DCR(R4) ; Reset DR32
                    00000800 8F    CA   009E   429          BICL    #DR_UTL_M_VALID,-       ; Clear WCS valid bit
                          04 A4         00A4   430                  DR_UTL(R4)
                                        00A6   431          ENBINT                          ; Lower IPL
                                        00A9   432
                                        00A9   433          ; Load last data rate saved in device dependent characteristics.
                                        00A9   434
                     50    04 A4    D0   00A9   435          MOVL    DR_UTL(R4),R0           ; Get contents of Utility reg.
                     50    44 A5    90   00AD   436          MOVB    UCB$L_DEVDEPEND(R5),R0  ; Load data rate
                     04 A4    50    D0   00B1   437          MOVL    R0,DR_UTL(R4)           ; Put back into Utility reg.
                                        00B5   438
                                        00B5   439          ; Set up to load microcode.  R0 will contain address of microcode image.
                                        00B5   440          ; R1 will contain size of image in bytes.  Convert this to number
                                        00B5   441          ; of WCS words by dividing by 5 (each WCS word contains 5 bytes).
                                        00B5   442          ; R2 will be used to contain WCS address.
                                        00B5   443
                                        00B5   444          ASSUME  DR_WCSA_V_SEL EQ 0
                                        00B5   445          ASSUME  DR_WCSA_V_ADDR EQ 1
                                        00B5   446
                       50    59    7D   00B5   447          MOVQ    R9,R0                   ; Restore address and size in R0 and R1
                       51    05    C6   00B8   448          DIVL    #5,R1                   ; Convert bytes to number of WCS words.
                             71    13   00BB   449          BEQL    40$                     ; Load zero words
                             51    D7   00BD   450          DECL    R1                      ; Convert to address of highest word
                       51    02    C4   00BF   451          MULL    #2,R1                   ; Account for address being incr. by 2
                       5B    51    D0   00C2   452          MOVL    R1,R11                  ; Save number of WCS words
                             52    D4   00C5   453          CLRL    R2                      ; Start loading at zero
                                        00C7   454
                                        00C7   455 20$:      ; Load next WCS word.  Each word gets loaded in two parts.  First
                                        00C7   456           ; four bytes get loaded and then one byte gets loaded.
                                        00C7   457           ; On the DR750, it is necessary to set the WCS flag bit to
                                        00C7   458           ; distinguish between accessing WCS and local store.  This bit has
                                        00C7   459           ; no effect on the DR780.
                                        00C7   460
                80000000 8F    C9   00C7   461          BISL3   #DR_WCSA_M_WCS,-        ; Load WCS address
```

```
                08 A4    52          00CD   462                 R2,DR_WCSA(R4)
                0C A4    80    D0    00D0   463          MOVL   (R0)+,DR_WCSD(R4)        ; Load four bytes
      08 A4  52 80000001 8F    C9    00D4   464          BISL3  #DR_WCSA_M_WCS!-         ; Load WCS address and set SELECT bit
                                     00DD   465                  DR_WCSA_M_SEL,R2,DR_WCSA(R4)
                0C A4    80    9A    00DD   466          MOVZBL (R0)+,DR_WCSD(R4)       ; Load one byte
           FFE0 52  02    51    F1    00E1   467          ACBL   R1,#2,R2,20$            ; Repeat load loop
                                     00E7   468
                                     00E7   469          ; Verify WCS by addressing each word (look for parity error later).
                                     00E7   470
                                     00E7   471          ASSUME DR_DCR  EQ  0
                                     00E7   472          ASSUME DR_UTL  EQ  DR_DCR+4
                                     00E7   473          ASSUME DR_UTL_M_PARERR  EQ  DR_DCR_M_PARERR
                                     00E7   474
                   50    54    D0    00E7   475          MOVL   R4,R0                   ; Get address of DR750's register with
                                     00EA   476                                         ; parity error bit (DR_DCR)
             03 68 A5    03    E0    00EA   477          BBS    #UCB_V_DR750,UCB$W_DEVSTS(R5),22$       ; Branch if DR750
                   50    04    C0    00EF   478          ADDL   #4,R0                   ; Get address of DR780's register with
                                     00F2   479                                         ; parity error bit (DR_UTL)
          60    80000000 8F    C8    00F2   480 22$:     BISL   #DR_UTL_M_PARERR,(R0)   ; Clear parity error
                   52    D4    00F9   481          CLRL   R2                      ; Clear WCS address
                80000000 8F    C9    00FB   482 25$:     BISL3  #DR_WCSA_M_WCS,-        ; Load WCS address
                  08 A4    52          0101   483                 R2,DR_WCSA(R4)
                  51    0C A4    D0    0104   484          MOVL   DR_WCSD(R4),R1          ; This allowes parity errors to be seen
           FFED 52  02    5B    F1    0108   485          ACBL   R1T,#2,R2,25$           ; Repeat verify loop
                                     010E   486
                                     010E   487          ; Reload if a powerfail occurred while loading.
                                     010E   488
                                     010E   489          DSBINT #31                     ; Raise IPL to lockout powerfail
             06 64 A5    05    E5    0114   490          BBCC   #UCB$V_POWER,UCB$W_STS(R5),30$  ; Branch if no powefail
                                     0119   491          ENBINT                         ; Powerfail occurred; lower IPL and
                      FF5D   31    011C   492          BRW    10$                     ; retry
                                     011F   493
                                     011F   494          ASSUME DR_UTL_V_PARERR  EQ  31
                                     011F   495
                   60    D5    011F   496 30$:     TSTL   (R0)                    ; No powerfail - Test for parity error
                   15    19    0121   497          BLSS   60$                     ; Branch if parity error
      04 A4  00000800 8F    C8    0123   498          BISL   #DR_UTL_M_VALID,DR_UTL(R4)  ; Set WCS valid
                                     012B   499          ENBINT                         ; Lower IPL
                                     012E   500
          64    0600 8F    3C    012E   501 40$:     MOVZWL #DCR_K_SETINTENB,DR_DCR(R4)  ; Set interrupt enable
                50    01    3C    0133   502          MOVZWL #SS$_NORMAL,R0          ; Return success status
                      08    11    0136   503          BRB    80$                     ; Return to user
                                     0138   504
                                     0138   505
                                     0138   506 60$:     ; ERROR - Parity error during WCS verification
                                     0138   507
                                     0138   508          ENBINT                         ; Lower IPL
          50    01F4 8F    3C    013B   509          MOVZWL #SS$_PARITY,R0          ; Completion code
                                     0140   510
                                     0140   511
                                     0140   512 80$:     ; Common return
                                     0140   513
           00000000'GF    17    0140   514          JMP    G^EXE$FINISHIOC         ; Finish I/O
```

```
0146   516          .SBTTL  STARTDATA_FDT - Start Data FDT routine
0146   517 ;++
0146   518 ; FUNCTIONAL DESCRIPTION:
0146   519 ;
0146   520 ;       This routine is the Start Data QIO FDT routine.  After it does
0146   521 ;       some error checking on the user's command table, it allocates
0146   522 ;       an I/O Request Packet Extension (IRPE) and links it onto the IRP.
0146   523 ;       Then the user's command block and buffer block are locked into
0146   524 ;       memory, with the context for these regions stored in the IRPE.
0146   525 ;       Finally, the address of the DR32's go bit is returned to the user
0146   526 ;       and the IRP is queued to the driver.
0146   527 ;
0146   528 ; CALLING SEQUENCE:
0146   529 ;
0146   530 ;       Called from the FDT routine dispatcher in the QIO system
0146   531 ;       service.  On completion, jumps to either EXE$QIODRVPKT (on success),
0146   532 ;       EXE$FINISHIOC (on errors that complete the I/O), or EXE$ABORTIO
0146   533 ;       (on errors that abort the I/O).
0146   534 ;
0146   535 ; INPUT PARAMETERS:
0146   536 ;
0146   537 ;       R3      Address of IRP
0146   538 ;       R4      Current process PCB address
0146   539 ;       R5      Address of UCB
0146   540 ;       R6      Address of CCB
0146   541 ;       P1(AP)  Address of command table
0146   542 ;       P2(AP)  Size (in bytes) of command table (must be XF$K_CMT_LENGTH)
0146   543 ;
0146   544 ; IMPLICIT INPUTS:
0146   545 ;
0146   546 ;       The format of the command table is:
0146   547 ;
0146   548 ;  P1(AP) ->   +-------------------------------+
0146   549 ;              |  Size of command block        |   XF$L_CMT_CBLKSZ
0146   550 ;              |  (in bytes)                   |
0146   551 ;              +-------------------------------+
0146   552 ;              |  Address of Command block     |   XF$L_CMT_CBLKAD
0146   553 ;              |  (Must be quadword aligned)   |
0146   554 ;              +-------------------------------+
0146   555 ;              |  Size of buffer block         |   XF$L_CMT_BBLKSZ
0146   556 ;              |  (in bytes)                   |
0146   557 ;              +-------------------------------+
0146   558 ;              |  Address of buffer block      |   XF$L_CMT_BBLKAD
0146   559 ;              |                               |
0146   560 ;              +-------------------------------+
0146   561 ;              |  Address of Packet AST        |   XF$L_CMT_PASTAD
0146   562 ;              |  routine                      |
0146   563 ;              +-------------------------------+
0146   564 ;              |  Packet AST parameter         |   XF$L_CMT_PASTPM
0146   565 ;              |                               |
0146   566 ;              +-------------------------------+
0146   567 ;              |               | Flags | Data  |   XF$B_CMT_RATE
0146   568 ;              |               |       | Rate  |   XF$B_CMT_FLAGS
0146   569 ;              +-------------------------------+
0146   570 ;              |  Address of longword to       |   XF$L_CMT_GBITAD
0146   571 ;              |  receive address of GO bit    |
0146   572 ;              +-------------------------------+
```

```
                     0146   573 ;
                     0146   574 ; OUTPUT PARAMETERS:
                     0146   575 ;
                     0146   576 ;       R0      Completion code (error returns only)
                     0146   577 ;
                     0146   578 ; IMPLICIT OUTPUTS:
                     0146   579 ;
                     0146   580 ;       None
                     0146   581 ;
                     0146   582 ; COMPLETION CODES:
                     0146   583 ;
                     0146   584 ;       Returned to EXE$ABORTIO:
                     0146   585 ;
                     0146   586 ;       SS$_ACCVIO      Access violation
                     0146   587 ;       SS$_INSFMEM     Insufficient dynamic memory
                     0146   588 ;
                     0146   589 ;       Returned to EXE$FINISHIOC:
                     0146   590 ;
                     0146   591 ;       SS$_IVBUFLEN    Invalid buffer length
                     0146   592 ;       SS$_BUFNOTALIGN Buffer not aligned correctly
                     0146   593 ;
                     0146   594 ; SIDE EFFECTS:
                     0146   595 ;
                     0146   596 ;       None
                     0146   597 ;--
                     0146   598
                     0146   599 STARTDATA_FDT:
                     0146   600
                     0146   601         ; First verify that the command table is accessible and of the
                     0146   602         ; correct length (XF$K_CMT_LENGTH bytes).
                     0146   603
          50   6C   7D  0146   604         MOVQ    P1(AP),R0                       ; Get address of command table in R0,
                     0149   605                                                 ; length in R1.
       5B  034C 8F   3C  0149   606         MOVZWL  #SS$_IVBUFLEN,R11               ; Assume error
           20   51   D1  014E   607         CMPL    R1,#XF$K_CMT_LENGTH             ; Is length correct?
                49   12  0151   608         BNEQ    FINISH_IO                       ; No, finish I/O
                     0153   609         IFNORD  R1,(R0),ACCESS_VIO              ; Br. if command table is not accessible
                     0159   610
                     0159   611         ; Copy command table onto stack so that it can't change (via I/O, for
                     0159   612         ; instance) while this FDT routine executes.
                     0159   613
          5E   51   C2  0159   614         SUBL    R1,SP                           ; Allocate space on stack
               38   BB  015C   615         PUSHR   #^M<R3,R4,R5>                   ; Save some registers
    0C AE  60   51   28  015E   616         MOVC3   R1,(R0),12(SP)                  ; Copy command table onto stack
               38   BA  0163   617         POPR    #^M<R3,R4,R5>                   ; Restore registers
                     0165   618
                     0165   619         ; Now check that the command block and buffer block sizes are
                     0165   620         ; greater than 0 and less than 2**29, that the command block is
                     0165   621         ; quadword aligned, and that the location to store the GO bit
                     0165   622         ; address is accessible.
                     0165   623
               6E   D5  0165   624         TSTL    XF$L_CMT_CBLKSZ(SP)             ; Is command block size zero?
               33   13  0167   625         BEQL    FINISH_IO                       ; Yes, finish I/O
               6E   D1  0169   626         CMPL    XF$L_CMT_CBLKSZ(SP),-           ; Is command block greater than or
 20000000 8F        016B   627                 #^X20000000                         ; equal to 2**29?
               2A   1E  0170   628         BGEQU   FINISH_IO                       ; Yes, finish I/O
         08 AE   D5  0172   629         TSTL    XF$L_CMT_BBLKSZ(SP)             ; Is buffer block size zero?
```

```
                     25   13  0175   630          BEQL    FINISH_IO                       ; Yes, finish I/O
                  08 AE   D1  0177   631          CMPL    XF$L_CMT_BBLKSZ(SP),-           ; Is buffer block greater than or
              20000000 8F       017A   632                  #^X20000000                     ; equal to 2**29?
                     1B   1E  017F   633          BGEQU   FINISH_IO                       ; Yes, finish I/O
           5B    0324 8F   3C  0181   634          MOVZWL  #SS$_BUFNOTALIGN,R11            ; Change completion code in R11
             04 AE    07   D3  0186   635          BITL    #7,XF$L_CMT_CBLKAD(SP)          ; Is command block quadword aligned?
                     10   12  018A   636          BNEQ    FINISH_IO                       ; No, finish I/O
                          018C   637          IFWRT   #4,@XF$L_CMT_GBITAD(SP),-; Br. if location to store GO bit
                          018C   638                  ALL_OK                          ; address is writeable
                          0193   639
                          0193   640          ;
                          0193   641          ; Error returns
                          0193   642          ;
                          0193   643
                          0193   644  ACCESS_VIO:
                  50   0C   3C  0193   645          MOVZWL  #SS$_ACCVIO,R0                  ; Store completion code
                          0196   646
                          0196   647  ABORT_IO:    ; Come here with completion code in R0
                          0196   648
              00000000'GF   17  0196   649          JMP     G^EXE$ABORTIO                   ; Abort I/O
                          019C   650
                          019C   651  FINISH_IO:  ; Come here with completion code in R11
                          019C   652
                  50   5B   D0  019C   653          MOVL    R11,R0                          ; Move completion code
              00000000'GF   17  019F   654          JMP     G^EXE$FINISHIOC                 ; Finish I/O
                          01A5   655
                          01A5   656
                          01A5   657  ALL_OK: ; Everything checks out.  Allocate an IRPE and link it to
                          01A5   658          ; the IRP.
                          01A5   659
                     53   DD  01A5   660          PUSHL   R3                              ; Save address of IRP
                00000000'GF   16  01A7   661          JSB     G^EXE$ALLOCIRP                  ; Allocate an IRPE (returns addr. in R2)
                     53 8ED0  01AD   662          POPL    R3                              ; Restore address of IRP
                   E3 50   E9  01B0   663          BLBC    R0,ABORT_IO                     ; Failed do to insufficient memory
                0A A2   2C   90  01B3   664          MOVB    #DYN$C_IRPE,IRPE$B_TYPE(R2)     ; Change type from IRP to IRPE
                54 A3   52   D0  01B7   665          MOVL    R2,IRP$L_EXTEND(R3)             ; Link IRPE onto IRP
           2A A3    0800 8F   A8  01BB   666          BISW    #IRP$M_EXTEND,IRP$W_STS(R3)     ; Set extend bit in status word
                   2A A2   B4  01C1   667          CLRW    IRPE$W_STS(R2)                  ; Clear status bits in IRPE
                   2C A2   D4  01C4   668          CLRL    IRPE$L_SVAPTE1(R2)              ; Clear SVAPTE for region 1
                   38 A2   D4  01C7   669          CLRL    IRPE$L_SVAPTE2(R2)              ; Clear SVAPTE for region 2
                5A   52   D0  01CA   670          MOVL    R2,R10                          ; From now on R10 points to IRPE
                          01CD   671
                          01CD   672          ; Now lock command block and buffer block into memory, saving
                          01CD   673          ; the context (SVAPTE, BCNT, and BOFF) in the IRPE.  R3 points to the
                          01CD   674          ; IRP, SP points to the command table, and R10 points to the IRPE.
                          01CD   675
                          01CD   676          ASSUME  IRPE$W_BOFF1  EQ  IRPE$L_SVAPTE1+4
                          01CD   677          ASSUME  IRPE$W_BOFF2  EQ  IRPE$L_SVAPTE2+4
                          01CD   678
                51   8E   D0  01CD   679          MOVL    (SP)+,R1                        ; Get length of command block
                50   8E   D0  01D0   680          MOVL    (SP)+,R0                        ; Get address of command block
             34 AA   51   D0  01D3   681          MOVL    R1,IRPE$L_BCNT1(R10)            ; Store length of command block in IRPE
             44 AA   50   D0  01D7   682          MOVL    R0,IRPE$L_CBLKADR(R10)          ; Store address of command block in IRPE
                   38   10  01DB   683          BSBB    LOCK_BFR                        ; Lock command block into memory
                2C A3   7D  01DD   684          MOVQ    IRP$L_SVAPTE(R3),-              ; Store SVAPTE and BOFF in IRPE
                2C AA       01E0   685                  IRPE$L_SVAPTE1(R10)
                51   8E   D0  01E2   686          MOVL    (SP)+,R1                        ; Get length of buffer block
```

```
                50    8E    D0   01E5    687          MOVL    (SP)+,R0                        ; Get address of buffer block
            40  AA    51    D0   01E8    688          MOVL    R1,IRP$L_BCNT2(R10)             ; Store length of buffer block in IRPE
            48  AA    50    D0   01EC    689          MOVL    R0,IRP$L_BUFLKADR(R10)          ; Store addr. of buffer block in IRPE
                  23  10    01F0    690          BSBB    LOCK_BFR                        ; Lock buffer block into memory
                2C  A3  7D    01F2    691          MOVQ    IRP$L_SVAPTE(R3),-              ; Store SVAPTE and BOFF in IRPE
                38  AA        01F5    692                  IRP$L_SVAPTE2(R10)
                2C  A3  7C    01F7    693          CLRQ    IRP$L_SVAPTE(R3)                ; Clear SVAPTE, BOFF, and BCNT in IRP
                              01FA    694
                              01FA    695          ; Now copy packet AST address and parameter into IRP and
                              01FA    696          ; return address of GO bit to user.
                              01FA    697
                              01FA    698          ASSUME  IRP$L_PKTASTPRM EQ IRP$L_PKTASTADR+4
                              01FA    699          ASSUME  IDB$L_CSR  EQ  0
                              01FA    700
            40  A3    8E    7D    01FA    701          MOVQ    (SP)+,IRP$L_PKTASTADR(R3) ; Store packet AST address and
                              01FE    702                                                  ; parameter in IRP
            3C  A3    8E    D0    01FE    703          MOVL    (SP)+,IRP$B_RATE(R3)      ; Store data rate and flags in IRP
            51    24  A5    D0    0202    704          MOVL    UCB$L_CRB(R5),R1          ; Get address of CRB in R1
         00000200   8F    C1    0206    705          ADDL3   #DR_USER,-                ; Store address of GO bit
            9E    2C  B1        020C    706                  @CRB$L_INTD+VEC$L_IDB(R1),@(SP)+
                              020F    707
      00000000'GF       17    020F    708          JMP     G^EXE$QIODRVPKT           ; Queue packet to driver
```

```
                     0215   710          .SBTTL  LOCK_BFR - Lock a buffer into memory
                     0215   711
                     0215   712 ;++
                     0215   713 ; FUNCTIONAL DESCRIPTION:
                     0215   714 ;
                     0215   715 ;       This routine is called from the Start Data FDT routine to lock
                     0215   716 ;       the command block and the buffer block into memory.  If either
                     0215   717 ;       lock fails, this routine unlocks any locked memory and deallocates
                     0215   718 ;       the IRPE before during a coroutine return to EXE$MODIFYLOCKR.
                     0215   719 ;
                     0215   720 ; CALLING SEQUENCE:
                     0215   721 ;
                     0215   722 ;       BSBB    LOCK_BFR        Note that if the lock fails, this routine
                     0215   723 ;                               returns to EXE$MODIFYLOCKR, not the caller.
                     0215   724 ;
                     0215   725 ; INPUT PARAMETERS:
                     0215   726 ;
                     0215   727 ;       R0      Address of buffer to lock
                     0215   728 ;       R1      Length of buffer (in bytes)
                     0215   729 ;       R3      Address of IRP
                     0215   730 ;       R4      Current process PCB address
                     0215   731 ;       R5      Address of UCB
                     0215   732 ;       R6      Address of CCB
                     0215   733 ;       R10     Address of IRPE
                     0215   734 ;
                     0215   735 ; IMPLICIT INPUTS:
                     0215   736 ;
                     0215   737 ;       Offsets IRPE$L_SVAPTE1, IRPE$W_BOFF1, and IRPE$L_BCNT1 in the
                     0215   738 ;       IRPE describe the previously locked area.
                     0215   739 ;
                     0215   740 ; OUTPUT PARAMETERS:
                     0215   741 ;
                     0215   742 ;       None (returning to the caller implies success)
                     0215   743 ;
                     0215   744 ; IMPLICIT OUTPUTS:
                     0215   745 ;
                     0215   746 ;       Offsets IRP$L_SVAPTE and IRP$W_BOFF in the IRP describe
                     0215   747 ;       the svapte and the byte offset of the locked area.
                     0215   748 ;
                     0215   749 ; COMPLETION CODES:
                     0215   750 ;
                     0215   751 ;       None
                     0215   752 ;
                     0215   753 ; SIDE EFFECTS:
                     0215   754 ;
                     0215   755 ;       As previously mentioned, on a lock failure the previously
                     0215   756 ;       locked area is unlocked, the IRPE is deallocated, and the I/O
                     0215   757 ;       is either completely backed up or aborted.
                     0215   758 ;--
                     0215   759
                     0215   760 LOCK_BFR:
00000000'GF   16     0215   761          JSB     G^EXE$MODIFYLOCKR       ; Lock buffer into memory
      35 50   E8     021B   762          BLBS    R0,90$                  ; Success!!!
                     021E   763
                     021E   764          ; Got a lock failure.  Unlock previously locked area if there is one.
                     021E   765
            0F   BB  021E   766          PUSHR   #^M<R0,R1,R2,R3>        ; Save registers
```

```
                2C A3    7C  0220   767          CLRQ    IRP$L_SVAPTE(R3)          ; Cleanup IRP so that we don't
                                    768                                            ; unlock same area twice
2A A3    0800 8F    AA   0223   769          BICW    #IRP$M_EXTEND,IRP$W_STS(R3)  ; Clear extend bit
      53    2C AA    D0   0229   770          MOVL    IRPE$L_SVAPTE1(R10),R3       ; Get SVAPTE
            19       13   022D   771          BEQL    10$                          ; No area previously locked
      51    30 AA    3C   022F   772          MOVZWL  IRPE$W_BOFF1(R10),R1         ; Get byte offset
      52    34 AA    D0   0233   773          MOVL    IRPE$L_BCNT1(R10),R2         ; Get byte count
  51    01FF C142    9E   0237   774          MOVAB   511(R1)[R2],R1               ; Combine offset and count and round
51    51    F7 8F    78   023D   775          ASHL    #-VA$S_BYTE,R1,R1            ; Convert to number of pages (to unlock)
    00000000'GF      16   0242   776          JSB     G^MMG$UNLOCK                 ; Unlock the pages
                              0248   777
                              0248   778 10$:  ; Now deallocate the IRPE.
                              0248   779
      50    5A       D0   0248   780          MOVL    R10,R0                       ; Address of packet
    00000000'GF      16   024B   781          JSB     G^EXE$DEANONPAGED            ; Deallocate it
            0F       BA   0251   782          POPR    #^M<R0,R1,R2,R3>             ; Restore registers
                              0253   783
                     05   0253   784 90$:    RSB                                  ; On success returns to caller,
                              0254   785                                          ; On failure returns to EXE$MODIFYLOCKR
```

```
                        0254   787          .SBTTL  STARTIO - Entry point to start I/O
                        0254   788
                        0254   789 ;++
                        0254   790 ; FUNCTIONAL DESCRIPTION:
                        0254   791 ;
                        0254   792 ;       This routine actually starts the DR32.  It loads the
                        0254   793 ;       required DR32 registers and clears the halt bit.
                        0254   794 ;
                        0254   795 ; CALLING SEQUENCE:
                        0254   796 ;
                        0254   797 ;       Jumped to through the driver dispatch table by IOC$INITIATE
                        0254   798 ;
                        0254   799 ; INPUT PARAMETERS:
                        0254   800 ;
                        0254   801 ;       R3      Address of the IRP
                        0254   802 ;       R5      Address of the UCB
                        0254   803 ;
                        0254   804 ; IMPLICIT INPUTS:
                        0254   805 ;
                        0254   806 ;       Various fields in the IRP, IRPE, and UCB.  In particular note
                        0254   807 ;       that offset UCB$L_IRP in the UCB contains the address of the IRP.
                        0254   808 ;
                        0254   809 ; OUTPUT PARAMETERS:
                        0254   810 ;
                        0254   811 ;       None
                        0254   812 ;
                        0254   813 ; IMPLICIT OUTPUTS:
                        0254   814 ;
                        0254   815 ;       None
                        0254   816 ;
                        0254   817 ; COMPLETION CODES:
                        0254   818 ;
                        0254   819 ;       Returned to REQ_COMPLETE:
                        0254   820 ;
                        0254   821 ;       SS$_POWERFAIL   Adapter has no power
                        0254   822 ;       SS$_MCNOTVALID  Microcode is not valid
                        0254   823 ;       SS$_BADPARAM    Specified data rate is too large
                        0254   824 ;
                        0254   825 ; SIDE EFFECTS:
                        0254   826 ;
                        0254   827 ;       None
                        0254   828 ;--
                        0254   829 STARTIO:
                        0254   830
                        0254   831          ; Get address of first CSR into R4.  Make sure adapter has power and
                        0254   832          ; then clear abort bit and parity error bit.
                        0254   833
                        0254   834          ASSUME  IDB$L_CSR  EQ  0
                        0254   835          ASSUME  UCB_V_ADPPWRUP  EQ  0
                        0254   836
          51   24 A5   D0  0254   837          MOVL    UCB$L_CRB(R5),R1                ; Get address of CRB
          54   2C B1   D0  0258   838          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R4    ; Get address of first CSR
            03 68 A5   E8  025C   839          BLBS    UCB$W_DEVSTS(R5),5$             ; Branch if adapter has power
                00EA   31  0260   840          BRW     40$                             ; Adapter has no power
        64    0400 8F   3C  0263   841 5$:      MOVZWL  #DCR_K_CLRABTINT,DR_DCR(R4)     ; Clear abort interrupt bit
00A0 C5     00040000 8F   CA  0268   842          BICL    #DR_DCR_M_DCRABT,UCB$L_DCR(R5)  ; Clear abort bit in DCR in UCB
  04 A4     80000000 8F   C8  0271   843          BISL    #DR_UTL_M_PARERR,DR_UTL(R4)     ; Clear parity error bit (DR780)
```

```
                64    80000000 8F    C8  0279    844          BISL    #DR_DCR_M_PARERR,DR_DCR(R4)     ; Clear parity error bit (DR750)
                                       0280    845
                                       0280    846          ; Load Utility register with data rate and parity error abort bit.
                                       0280    847
                                       0280    848          ASSUME  XF$V_CMT_SETRTE  EQ  0
                                       0280    849
                     50    04 A4    D0  0280    850          MOVL    DR_UTL(R4),R0           ; Read contents of Utility register
                50   08000000 8F    CA  0284    851          BICL    #DR_UTL_M_ENPEAB,R0     ; Clear enable par. err. abort bit (DR780)
                64   08000000 8F    CA  028B    852          BICL    #DR_DCR_M_ENPEAB,DR_DCR(R4) ; Clear enable par. err. abort (DR750)
                        28 3D A3    E9  0292    853          BLBC    IRP$B_FLAGS(R3),10$     ; Branch if we shouldn't set rate
                           3C A3    91  0296    854          CMPB    IRP$B_RATE(R3),-        ; Compare specified rate with maximum
                     00000000'GF            0299    855          G^IOC$GW_XFMXRATE       ; allowed by SYSGEN parameter
                                 13    1A  029E    856          BGTRU   8$                      ; Rate too high - error
                     51    FB 8F    9A  02A0    857          MOVZBL  #DR780_MAXRATE,R1       ; Get hardware maximum clock rate (DR780)
                   04 68 A5    03    E1  02A4    858          BBC     #UCB_V_DR750,UCB$W_DEVSTS(R5),7$  ; Branch if DR780
                     51    FC 8F    9A  02A9    859          MOVZBL  #DR750_MAXRATE,R1       ; Get hardware maximum clock rate (DR750)
                     51    3C A3    91  02AD    860  7$:     CMPB    IRP$B_RATE(R3),R1       ; Compare specified rate with maximum
                                 03    1B  02B1    861          BLEQU   9$                      ; allowed by hardware and branch if ok
                                 00A8  31  02B3    862  8$:     BRW     60$                     ; Rate too high - error
                     50    3C A3    90  02B6    863  9$:     MOVB    IRP$B_RATE(R3),R0       ; Set rate
                     44 A5    50    90  02BA    864          MOVB    R0,UCB$L_DEVDEPEND(R5)  ; Put rate into device characteristics
                                 01    E0  02BE    865  10$:    BBS     #XF$V_CMT_DIPEAB,-      ; Branch if we shouldn't set abort
                        0E 3D A3            02C0    866                  IRP$B_FLAGS(R3),20$     ; on parity error bit
                50   08000000 8F    C8  02C3    867          BISL    #DR_UTL_M_ENPEAB,R0     ; Set abort on parity error bit (DR780)
                64   08000000 8F    C8  02CA    868          BISL    #DR_DCR_M_ENPEAB,DR_DCR(R4)  ; Set abort on par. err. bit (DR750)
                     04 A4    50    D0  02D1    869  20$:    MOVL    R0,DR_UTL(R4)           ; Load Utility register
                                       02D5    870
                                       02D5    871          ; Load up the rest of the DR32 registers.
                                       02D5    872          ; Note that the DR780 registers are directly addressable while
                                       02D5    873          ; the DR750 registers are accessed by loading the register number
                                       02D5    874          ; into the DR_WCSA register and reading or writing the DR_WCSD register.
                                       02D5    875
                     52    54 A3    D0  02D5    876          MOVL    IRP$L_EXTEND(R3),R2     ; R2 points to IRPE
                   29 68 A5    03    E0  02D9    877          BBS     #UCB_V_DR750,UCB$W_DEVSTS(R5),30$  ; Branch if DR750
                                       02DE    878
                                       02DE    879          ; Load DR780 registers
                                       02DE    880
                     51    0404 C4    DE  02DE    881          MOVAL   DR_780_SBR(R4),R1       ; R1 will step through registers
                                       02E3    882
                                       02E3    883
                           81    0C    DB  02E3    884          MFPR    #PR$_SBR,(R1)+          ; DR_SBR = contents of sys. base reg.
                81   00000000'GF    D0  02E6    885          MOVL    G^MMG$GL_GPTBASE,(R1)+  ; DR_GBR = address of global page table
                     81    44 A2    D0  02ED    886          MOVL    IRPE$L_CBLKADR(R2),(R1)+; DR_CMDBVA = address of command block
                     81    34 A2    D0  02F1    887          MOVL    IRPE$L_BCNT1(R2),(R1)+  ; DR_CMDLEN = length of command block
                     81    2C A2    D0  02F5    888          MOVL    IRPE$L_SVAPTE1(R2),(R1)+; DR_CMDSVAPTE = SVAPTE of command block
                     81    48 A2    D0  02F9    889          MOVL    IRPE$L_BBLKADR(R2),(R1)+; DR_BFRBVA = address of buffer block
                     81    40 A2    D0  02FD    890          MOVL    IRPE$L_BCNT2(R2),(R1)+  ; DR_BFRLEN = length of buffer block
                     81    38 A2    D0  0301    891          MOVL    IRPE$L_SVAPTE2(R2),(R1)+; DR_BFRSVAPTE = SVAPTE of buffer block
                                 2D    11  0305    892          BRB     35$
                                       0307    893
                                       0307    894  30$:    ; Load DR750 registers
                                       0307    895
                     51    01    D0  0307    896          MOVL    #DR_750_SBR,R1          ; R1 will step through registers
                     50    0C A4    DE  030A    897          MOVAL   DR_WCSD(R4),R0          ; R0 will point to WCS data register
                                       030E    898
                     08 A4    81    9E  030E    899          MOVAB   (R1)+,DR_WCSA(R4)       ; Store address of next register
                              60    0C    DB  0312    900          MFPR    #PR$_SBR,(R0)           ; DR_SBR = contents of sys. base reg.
```

```
                60   00000000'GF   D0   0315   901          MOVL    G^MMG$GL_GPTBASE,(R0)    ; DR_GBR = address of global page table
                  60      44 A2     D0   031C   902          MOVL    IRPE$L_CBLKADR(R2),(R0)  ; DR_CMDBVA = address of command block
                  60      34 A2     D0   0320   903          MOVL    IRPE$L_BCNT1(R2),(R0)    ; DR_CMDLEN = length of command block
                  60      2C A2     D0   0324   904          MOVL    IRPE$L_SVAPTE1(R2),(R0)  ; DR_CMDSVAPTE = SVAPTE of command block
                  60      48 A2     D0   0328   905          MOVL    IRPE$L_BBLKADR(R2),(R0)  ; DR_BFRBVA = address of buffer block
                  60      40 A2     D0   032C   906          MOVL    IRPE$L_BCNT2(R2),(R0)    ; DR_BFRLEN = length of buffer block
                  60      38 A2     D0   0330   907          MOVL    IRPE$L_SVAPTE2(R2),(R0)  ; DR_BFRSVAPTE = SVAPTE of buffer block
                                           0334   908
                                           0334   909 35$:    ; Check for WCS valid and start the DR going!
                                           0334   910
                                           0334   911          DSBINT  #31                      ; Raise IPL to 31
            04 A4    00000800 8F     D3   033A   912          BITL    #DR_UTL_M_VALID,DR_UTL(R4) ; Is WCS valid?
                           10        13   0342   913          BEQL    50$                      ; No, error!
                  64     0700 8F     3C   0344   914          MOVZWL  #DCR_K_CLRHLT,DR_DCR(R4) ; Clear Halt function
                                           0349   915          ENBINT                           ; Lower IPL
                                     05    034C   916          RSB
                                           034D   917
                                           034D   918
                                           034D   919
                                           034D   920 40$:    ; Error - Adapter has no power
                                           034D   921
                  50     0364 8F     3C   034D   922          MOVZWL  #SS$_POWERFAIL,R0        ; Status
                                     0D    11   0352   923          BRB     70$
                                           0354   924
                                           0354   925 50$:    ; Error - WCS not valid
                                           0354   926
                                           0354   927          ENBINT                           ; Lower IPl
                  50     035C 8F     3C   0357   928          MOVZWL  #SS$_MCNOTVALID,R0       ; Status
                                     03    11   035C   929          BRB     70$
                                           035E   930
                                           035E   931 60$:    ; Error - Data rate too high
                                           035E   932
               50     14     3C   035E   933          MOVZWL  #SS$_BADPARAM,R0         ; Status
                                           0361   934
                                           0361   935 70$:    ; Complete I/O with error in R0
                                           0361   936
                     51     D4   0361   937          CLRL    R1                       ; Clear second half of I/O status block
                  0136      31   0363   938          BRW     REQ_COMPLETE
```

```
                     0366   940          .SBTTL  INTERRUPT_SVC - Interrupt service routine
                     0366   941
                     0366   942 ;++
                     0366   943 ; FUNCTIONAL DESCRIPTION:
                     0366   944 ;
                     0366   945 ;       This is the interrupt service routine for DR32 interrupts.
                     0366   946 ;       It reads the DCR register, clears the interrupting condition(s),
                     0366   947 ;       and calls HANDLE_INT to handle the interrupt.  Note that this
                     0366   948 ;       routine executes at device IPL while HANDLE_INT forks and
                     0366   949 ;       therefore executes at fork IPL.
                     0366   950 ;
                     0366   951 ; CALLING SEQUENCE:
                     0366   952 ;
                     0366   953 ;       JSB from interrupt vector in CRB.  This routine cleans up
                     0366   954 ;       the stack and does an REI.
                     0366   955 ;
                     0366   956 ; INPUT PARAMETERS:
                     0366   957 ;
                     0366   958 ;       None
                     0366   959 ;
                     0366   960 ; IMPLICIT INPUTS:
                     0366   961 ;
                     0366   962 ;       The stack on entry is as follows:
                     0366   963 ;
                     0366   964 ;               0(SP)           Address of IDB address
                     0366   965 ;               4(SP)           Saved R2
                     0366   966 ;               8(SP)           Saved R3
                     0366   967 ;              12(SP)           Saved R4
                     0366   968 ;              16(SP)           Saved R5
                     0366   969 ;              20(SP)           Interrupt PC
                     0366   970 ;              24(SP)           Interrupt PSL
                     0366   971 ;
                     0366   972 ; OUTPUT PARAMETERS:
                     0366   973 ;
                     0366   974 ;       None
                     0366   975 ;
                     0366   976 ; IMPLICIT OUTPUTS:
                     0366   977 ;
                     0366   978 ;       None
                     0366   979 ;
                     0366   980 ; COMPLETION CODES:
                     0366   981 ;
                     0366   982 ;       None
                     0366   983 ;
                     0366   984 ; SIDE EFFECTS:
                     0366   985 ;
                     0366   986 ;       None
                     0366   987 ;--
                     0366   988
                     0366   989          ASSUME  IDB$L_CSR+4  EQ  IDB$L_OWNER
                     0366   990          ASSUME  DR_DCR_V_PWR_DN  EQ  DR_DCR_V_PWR_UP+1
                     0366   991
                     0366   992 INTERRUPT_SVC:
53   9E   D0          0366   993          MOVL    @(SP)+,R3               ; Get address of IDB in R3
54   63   7D          0369   994          MOVQ    IDB$L_CSR(R3),R4        ; CSR -> R4,  UCB-> R5
                     036C   995
                     036C   996          ; Get contents of DCR and OR into DCR in UCB
```

```
                                036C   997
                 53   64   D0  036C   998          MOVL    DR_DCR(R4),R3           ; Get DCR
            00A0 C5   53   C8  036F   999          BISL    R3,UCB$L_DCR(R5)        ; OR into DCR in UCB for use after FORK
                                0374  1000
                                0374  1001
                                0374  1002          ; Clear interrupting conditions.  Note that if there is a
                                0374  1003          ; power up/down interrupt, then those bits are jammed into
                                0374  1004          ; the DCR in the UCB.
                                0374  1005
                  05 53   13   E1  0374  1006          BBC     #DR_DCR_V_PKTINT,R3,10$ ; Branch if not packet interrupt
                64   3000 8F   3C  0378  1007          MOVZWL  #DCR_K_CLRPKTINT,DR_DCR(R4)  ; Clear packet interrupt
                  05 53   12   E1  037D  1008 10$:     BBC     #DR_DCR_V_DCRABT,R3,20$ ; Branch if not abort interrupt
                64   0400 8F   3C  0381  1009          MOVZWL  #DCR_K_CLRABTINT,DR_DCR(R4)  ; Clear abort interrupt
                     52   64   D0  0386  1010 20$:     MOVL    DR_DCR(R4),R2           ; Get DCR
                  12 52   11   E1  0389  1011          BBC     #DR_DCR_V_DCRHLT,R2,26$ ; Br if halt not set
              52   000C0000 8F   D3  038D  1012          BITL    #DR_DCR_M_PKTINT !-
                                0394  1013                    DR_DCR_M_DCRABT,R2      ; Is one of these bits set?
                           09   12  0394  1014          BNEQ    26$                     ; Br if set
            00A0 C5   00020000 8F   C8  0396  1015          BISL    #DR_DCR_M_DCRHLT,UCB$L_DCR(R5) ; Else set the halt bit
              53   00C00000 8F   D3  039F  1016 26$:     BITL    #DR_DCR_M_PWR_UP!-      ; Power up or power down bit set?
                                03A6  1017                    DR_DCR_M_PWR_DN,R3
                           1E   13  03A6  1018          BEQL    40$                     ; No.
                     02    16   EF  03A8  1019          EXTZV   #DR_DCR_V_PWR_UP,#2,-   ; Extract power up and power down bits
                     52    53       03AB  1020                  R3,R2
                     16    52   F0  03AD  1021          INSV    R2,#DR_DCR_V_PWR_UP,-   ; Insert them into DCR in UCB
                00A0 C5    02       03B0  1022                  #2,UCB$L_DCR(R5)
                  05 53    16   E1  03B4  1023          BBC     #DR_DCR_V_PWR_UP,R3,30$ ; Branch if not power up interrupt
                64   0100 8F   3C  03B8  1024          MOVZWL  #DCR_K_CLRPWRUP,DR_DCR(R4)  ; Clear power up interrupt
                  05 53    17   E1  03BD  1025 30$:     BBC     #DR_DCR_V_PWR_DN,R3,40$ ; Branch if not power down interrupt
                64   0200 8F   3C  03C1  1026          MOVZWL  #DCR_K_CLRPWRDN,DR_DCR(R4)  ; Clear power down interrupt
                                03C6  1027
                                03C6  1028 40$:     ; Test and set interlock bit.  Purpose of interlock bit is to
                                03C6  1029          ; prevent FORKing while the FORK block is in use as a result of
                                03C6  1030          ; a previous interrupt.
                                03C6  1031
                     01   E2   03C6  1032          BBSS    #UCB_V_FKLOCK,-         ; Branch if interlock is set
              02 68 A5           03C8  1033                  UCB$W_DEVSTS(R5),50$
                                03CB  1034
                     07   10   03CB  1035          BSBB    HANDLE_INT              ; Handle interrupt (at fork IPL)
                                03CD  1036
                                03CD  1037 50$:     ; Finish cleaning up stack and return from interrupt
                                03CD  1038
                52   8E   7D   03CD  1039          MOVQ    (SP)+,R2                ; Restore R2 and R3
                54   8E   7D   03D0  1040          MOVQ    (SP)+,R4                ; Restore R4 and R5
                           02   03D3  1041          REI
```

```
                        1043          .SBTTL  HANDLE_INT - Handle the interrupt
                        1044
                        1045 ;++
                        1046 ; FUNCTIONAL DESCRIPTION:
                        1047 ;
                        1048 ;       This routine actually handles the interrupts.  It is called
                        1049 ;       by INTERRUPT_SVC at device IPL but immediately forks to fork IPL.
                        1050 ;       There are four interrupting conditions:  power up, power down,
                        1051 ;       abort, and packet interrupts.  This routine checks for all of these
                        1052 ;       conditions.  If there is an I/O in progress, there are three possible
                        1053 ;       exits from this routine:  If the DR is halted, then the I/O is
                        1054 ;       completed (with either a success or failure status code).  If the
                        1055 ;       DR is not halted, then if no errors were detected the transfer
                        1056 ;       continues.  If any errors were detected, the transfer is aborted.
                        1057 ;
                        1058 ; CALLING SEQUENCE:
                        1059 ;
                        1060 ;       BSBB from interrupt service routine
                        1061 ;
                        1062 ; INPUT PARAMETERS:
                        1063 ;
                        1064 ;       R4      Address of first device CSR
                        1065 ;       R5      Address of UCB
                        1066 ;
                        1067 ; IMPLICIT INPUTS:
                        1068 ;
                        1069 ;       Offset UCB$L_DCR in the UCB contains the device DCR register
                        1070 ;
                        1071 ; OUTPUT PARAMETERS:
                        1072 ;
                        1073 ;       None
                        1074 ;
                        1075 ; IMPLICIT OUTPUTS:
                        1076 ;
                        1077 ;       None
                        1078 ;
                        1079 ; COMPLETION CODES:
                        1080 ;
                        1081 ;       None
                        1082 ;
                        1083 ; SIDE EFFECTS:
                        1084 ;
                        1085 ;       None
                        1086 ;--
                        1087
                        1088 HANDLE_INT:
                        1089          IOFORK
                        1090
                        1091          ; Clear FORK interlock, get DCR from UCB, and clear DCR in UCB.
                        1092          ; This clear must be performed by a BICL as it is possible for
                        1093          ; the interrupt service routine to OR in additional bits
                        1094          ; between the get DCR and the clear DCR.  These additional bits
                        1095          ; will be handled by the next interrupt.
                        1096
                        1097          SETIPL  UCB$B_DIPL(R5)          ; Raise IPL to prevent race condition
    02      AA          1098          BICW    #UCB_M_FKLOCK,-         ; Clear fork interlock
  68 A5                 1099                  UCB$W_DEVSTS(R5)
```

```
                                ;
                                ; Now test for cause of interrupt.  The possibilities are:
                                ;       1)  Power up interrupt
                                ;       2)  Power down interrupt
                                ;       3)  Abort interrupt
                                ;       4)  Packet interrupt
                                ;
                                ; Test for power up interrupt
                                ;
52   00A0 C5   D0  03E2  1100           MOVL    UCB$L_DCR(R5),R2        ; Get DCR from UCB
00A0 C5   52   CA  03E7  1101           BICL    R2,UCB$L_DCR(R5)        ; Clear DCR in UCB
                   03EC  1102           SETIPL  UCB$B_FIPL(R5)          ; Lower to fork IPL
00A8 C5   52   D0  03F0  1103           MOVL    R2,UCB$L_SAVDCR(R5)     ; Save DCR for reg. dump routine
                   03F5  1104
  53   58 A5   D0  03F5  1105           MOVL    UCB$L_IRP(R5),R3        ; Get address of IRP (if there is one)
       50 01   3C  03F9  1106           MOVZWL  #SS$_NORMAL,R0          ; Assume success
          51   D4  03FC  1107           CLRL    R1                      ; Clear second half of I/O status block
                   03FE  1108
                   03FE  1109           ; Now test for cause of interrupt.  The possibilities are:
                   03FE  1110           ;       1)  Power up interrupt
                   03FE  1111           ;       2)  Power down interrupt
                   03FE  1112           ;       3)  Abort interrupt
                   03FE  1113           ;       4)  Packet interrupt
                   03FE  1114
                   03FE  1115           ; Test for power up interrupt
                   03FE  1116
  0D 52   16   E1  03FE  1117           BBC     #DR_DCR_V_PWR_UP,R2,20$ ; Branch if not power up interrupt
  68 A5   01   A8  0402  1118           BISW    #UCB_M_ADPPWRUP,UCB$W_DEVSTS(R5)  ; Set adapter power up bit
  05 52   11   E1  0406  1119           BBC     #DR_DCR_V_DCRHLT,R2,20$ ; Branch if device is not halted
50   0364 8F   3C  040A  1120           MOVZWL  #SS$_POWERFAIL,R0       ; Load powerfail status
                   040F  1121
                   040F  1122 20$:     ; Test for power down interrupt
                   040F  1123
  0D 52   17   E1  040F  1124           BBC     #DR_DCR_V_PWR_DN,R2,30$ ; Branch if not power down interrupt
  68 A5   01   AA  0413  1125           BICW    #UCB_M_ADPPWRUP,UCB$W_DEVSTS(R5)  ; Clear adapter power up bit
  64 A5   20   A8  0417  1126           BISW    #UCB$M_POWER,UCB$W_STS(R5)  ; Set powerfail bit in UCB
50   0364 8F   3C  041B  1127           MOVZWL  #SS$_POWERFAIL,R0       ; Load powerfail status
                   0420  1128
                   0420  1129 30$:     ; Skip remaining code if we don't have an IRP (UCB is not busy).
                   0420  1130
       08   E1  0420  1131           BBC     #UCB$V_BSY,-            ; Branch if we don't have an IRP
 49 64 A5         0422  1132                   UCB$W_STS(R5),DONE
                   0425  1133
                   0425  1134           ; Test for Abort Interrupt
                   0425  1135
03 52   12   E1  0425  1136           BBC     #DR_DCR_V_DCRABT,R2,40$ ; Branch if not abort interrupt
     0086  30   0429  1137           BSBW    ABORT_INT               ; Handle abort interrupt
                   042C  1138
                   042C  1139 40$:     ; Test for packet interrupt
                   042C  1140
0F 52   13   E1  042C  1141           BBC     #DR_DCR_V_PKTINT,R2,60$ ; Branch if not packet interrupt
     3F   BB  0430  1142           PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Have packet int.  Save registers.
   00F6   30  0432  1143           BSBW    QUEUE_PKT_AST           ; Queue packet AST
05 50   E8  0435  1144           BLBS    R0,50$                  ; Success
     51   D4  0438  1145           CLRL    R1                      ; Failure to queue AST
  6E 50   7D  043A  1146           MOVQ    R0,(SP)                 ; Store status
     3F   BA  043D  1147 50$:     POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
                   043F  1148
                   043F  1149 60$:     ; Come here with I/O status block in R0 and R1.  If DR is halted,
                   043F  1150          ; send IRP to request complete.  If the DR is not halted, return
                   043F  1151          ; if status is success and abort the DR if the status is failure
                   043F  1152          ; or if an abort is pending.
                   043F  1153
59 52   11   E0  043F  1154           BBS     #DR_DCR_V_DCRHLT,R2,-   ; Branch if device is halted
                   0443  1155                   REQ_COMPLETE            ; to request complete
     02   E0  0443  1156           BBS     #UCB_V_ABORT,-          ; Branch if an abort is pending
```

```
                08 68 A5        0445  1157                    UCB$W_DEVSTS(R5),WAIT
                   23 50   E8   0448  1158            BLBS    R0,DONE                 ; If status is success, return
                                044B  1159
                                044B  1160 ABORT:   ; Abort the device by setting ext. abort.  Save the status in R0 in
                                044B  1161          ; the UCB for use when the device interrupts with an abort interrupt.
                                044B  1162          ; Note that this entry is called as a subroutine by CANCEL_IO
                                044B  1163
            00A4 C5   50    D0  044B  1164            MOVL    R0,UCB$L_SAVSTATUS(R5)  ; Save status in UCB
                                0450  1165 WAIT:      DSBINT                          ; Disable interrupts
            64   2000 8F    3C  0456  1166            MOVZWL  #DCR_K_SETEXTABT,DR_DCR(R4)  ; Abort the device
              68 A5   04    A8  045B  1167            BISW    #UCB_M_ABORT,UCB$W_DEVSTS(R5); Set the abort pending bit
                      01    E0  045F  1168            BBS     #UCB_V_FKLOCK,-         ; Don't wait for interrupt if
                08 68 A5        0461  1169                    UCB$W_DEVSTS(R5),DONE   ; we already have one
                                0464  1170            WFIKPCH TIMEOUT,#2              ; Wait for interrupt - 2 second timeout
                                046E  1171
                                046E  1172
                                046E  1173 DONE:    ; Common return
                                046E  1174
                                046E  1175            RSB
                                046F  1176
                                046F  1177
                                046F  1178 TIMEOUT:
                                046F  1179          ; Come here (at device IPL) on timeout or powerfail while
                                046F  1180          ; waiting for an abort interrupt.
                                046F  1181
            00A8 C5   64    D0  046F  1182            MOVL    DR_DCR(R4),UCB$L_SAVDCR(R5) ; Save DCR for reg. dump routine
            64   4000 8F    3C  0474  1183            MOVZWL  #DCR_K_RESET,DR_DCR(R4) ; Reset DR32
            64   0600 8F    3C  0479  1184            MOVZWL  #DCR_K_SETINTENB,DR_DCR(R4)  ; Enable interrupts
                                047E  1185            SETIPL  UCB$B_FIPL(R5)          ; Drop to fork IPL
             50  022C 8F    3C  0482  1186            MOVZWL  #SS$_TIMEOUT,R0         ; Assume timeout status
          05 64 A5   05    E1  0487  1187            BBC     #UCB$V_POWER,UCB$W_STS(R5),10$  ; Branch if not powerfail
             50  0364 8F    3C  048C  1188            MOVZWL  #SS$_POWERFAIL,R0       ; Set powerfail status
             51  00A4 C5    D0  0491  1189 10$:       MOVL    UCB$L_SAVSTATUS(R5),R1  ; Store saved status (reason for abort) in R
              00000000'GF    16  0496  1190            JSB     G^ERL$DEVICTMO          ; Error log timeout
                                049C  1191                                            ; Fall through to complete request
                                049C  1192
                                049C  1193
                                049C  1194 REQ_COMPLETE:
              68 A5    04    AA  049C  1195            BICW    #UCB_M_ABORT,UCB$W_DEVSTS(R5)  ; Clear abort pending bit
                 7E    50    7D  04A0  1196            MOVQ    R0,-(SP)                ; Save R0 and R1 (I/O status block)
              00000000'GF    16  04A3  1197            JSB     G^IOC$DIAGBUFILL        ; Fill diagnostic buffer (if specified)
                 50    8E    7D  04A9  1198            MOVQ    (SP)+,R0                ; Restore R0 and R1
                                04AC  1199            REQCOM                          ; Complete the IRP.  R0 and R1
                                04B2  1200                                            ; contain I/O status block.
```

```
04B2   1202          .SBTTL  ABORT_INT - Handle abort interrupts
04B2   1203
04B2   1204 ;++
04B2   1205 ; FUNCTIONAL DESCRIPTION:
04B2   1206 ;
04B2   1207 ;       This routine handles abort interrupts.  It distinguishes among
04B2   1208 ;       four cases.  These are:
04B2   1209 ;               1)  Driver abort
04B2   1210 ;               2)  User and far end device errors
04B2   1211 ;               3)  Parity errors
04B2   1212 ;               4)  Other DR32 errors (such as bus errors)
04B2   1213 ;       This routine's main purpose is to identify the cause of the abort
04B2   1214 ;       and to return appropriate status in R0 and R1 for use as the
04B2   1215 ;       I/O status block.
04B2   1216 ;       This routine also error logs parity errors and DR32 errors (cases
04B2   1217 ;       3 and 4 above).
04B2   1218 ;
04B2   1219 ; CALLING SEQUENCE:
04B2   1220 ;
04B2   1221 ;       BSBB    ABORT_INT
04B2   1222 ;
04B2   1223 ; INPUT PARAMETERS:
04B2   1224 ;
04B2   1225 ;       R2      Contents of DCR
04B2   1226 ;       R4      Address of first device CSR
04B2   1227 ;       R5      Address of UCB
04B2   1228 ;
04B2   1229 ; IMPLICIT INPUTS:
04B2   1230 ;
04B2   1231 ;       Offset UCB$L_SAVSTATUS in the UCB contains the status for driver aborts
04B2   1232 ;
04B2   1233 ; OUTPUT PARAMETERS:
04B2   1234 ;
04B2   1235 ;       R0      First longword of I/O status block
04B2   1236 ;       R1      Second longword of I/O status block
04B2   1237 ;
04B2   1238 ; IMPLICIT OUTPUTS:
04B2   1239 ;
04B2   1240 ;       None
04B2   1241 ;
04B2   1242 ; COMPLETION CODES:
04B2   1243 ;
04B2   1244 ;       R0      contains status left in UCB$L_SAVSTATUS for driver aborts,
04B2   1245 ;               SS$_DEVREQERR for user and far end device errors,
04B2   1246 ;               SS$_PARITY for parity errors, and
04B2   1247 ;               SS$_CTRLERR for other errors.
04B2   1248 ;
04B2   1249 ;       R1      0 for driver aborts, and
04B2   1250 ;               a combination of bits from the DCR, UTILITY, and DSL
04B2   1251 ;               registers for all other errors.
04B2   1252 ;
04B2   1253 ; SIDE EFFECTS:
04B2   1254 ;
04B2   1255 ;       None
04B2   1256 ;--
04B2   1257
04B2   1258 ABORT_INT:
```

```
                                04B2  1259          ; Test for Driver abort.
                                04B2  1260
          08 52    18   E1      04B2  1261          BBC     #DR_DCR_V_EXTABT,R2,10$ ; Branch if not driver abort
       50    00A4 C5   D0      04B6  1262          MOVL    UCB$L_SAVSTATUS(R5),R0  ; Get status saved when driver set abort
                  51   D4      04BB  1263          CLRL    R1                      ; Second longword of IOSB is 0
                       05      04BD  1264          RSB                             ; Return
                               04BE  1265
                               04BE  1266 10$:     ; Collect bits for second longword of I/O status block as all
                               04BE  1267          ; other errors require this.
                               04BE  1268
                               04BE  1269          ASSUME  DR_UTL_V_WCSPE  EQ  DR_DCR_V_WCSPE
                               04BE  1270          ASSUME  DR_750_DSL  EQ  0
                               04BE  1271
       07 68 A5    03   E0      04BE  1272          BBS     #UCB_V_DR750,UCB$W_DEVSTS(R5),20$  ; Branch if DR750
        51    0400 C4   D0      04C3  1273          MOVL    DR_780_DSL(R4),R1       ; Start out with all of the DSL
                  07   11      04C8  1274          BRB     30$
             08 A4    D4      04CA  1275 20$:     CLRL    DR_WCSA(R4)             ; Store register number to get DSL
          51    0C A4    D0      04CD  1276          MOVL    DR_WCSD(R4),R1          ; Start out with all of the DSL
        52    5500 8F   B3      04D1  1277 30$:     BITW    #DR_DCR_M_ID1ERR!-      ; OR these four bits into 1 bit
                               04D6  1278            DR_DCR_M_ID1TO!-        ; These are sort of bus errors
                               04D6  1279            DR_DCR_M_ID2ERR!-
                               04D6  1280            DR_DCR_M_ID2TO,R2
                  04   13      04D6  1281          BEQL    40$                     ; None of those bits were set
          00 51    1B   E2      04D8  1282          BBSS    #XF$V_IOS_BUSERR,R1,40$ ; Set bus error bit
          04 52    0F   E1      04DC  1283 40$:     BBC     #DR_DCR_V_RDS,R2,45$    ; Branch if Read Data Substitute not set
          00 51    1C   E2      04E0  1284          BBSS    #XF$V_IOS_RDSERR,R1,45$ ; Set Read Data Substitute error
          50    04 A4    D0      04E4  1285 45$:     MOVL    DR_UTL(R4),R0           ; Get Utility register (contains parity
                               04E8  1286                                          ; error bits on DR780)
       03 68 A5    03   E1      04E8  1287          BBC     #UCB_V_DR750,UCB$W_DEVSTS(R5),50$  ; Branch if DR780
          50    52   D0      04ED  1288          MOVL    R2,R0                   ; Get DCR (for parity err bits on DR750)
    50    50    E4 8F   78      04F0  1289 50$:     ASHL    #-DR_UTL_V_WCSPE,R0,R0  ; Shift 3 parity error bits to bit 0
             1D    50   F0      04F5  1290          INSV    R0,#XF$V_IOS_WCSPE,-    ; and insert in R1
             51    03           04F8  1291          #3,R1
                               04FA  1292
                               04FA  1293          ; Test for controller error (SS$_CTRLERR)
                               04FA  1294
    51    18000004 8F   D3      04FA  1295          BITL    #XF$M_IOS_BUSERR!-      ; Test for bus error,
                               0501  1296            XF$M_IOS_RDSERR!-       ; read data substitute error,
                               0501  1297            XF$M_IOS_INVPTE,R1      ; and invalid PTE
                  1C   12      0501  1298          BNEQ    60$                     ; Have a controller error
                               0503  1299
                               0503  1300          ; Test for Parity error (SS$_PARITY)
                               0503  1301
          50    01F4 8F   3C      0503  1302          MOVZWL  #SS$_PARITY,R0          ; Assume yes
    51    E0000000 8F   D3      0508  1303          BITL    #XF$M_IOS_WCSPE!-       ; Test for WCS parity error,
                               050F  1304            XF$M_IOS_CIPE!-         ; control interconnect parity error,
                               050F  1305            XF$M_IOS_DIPE,R1        ; and data interconnect parity error
                  13   12      050F  1306          BNEQ    70$                     ; Have a parity error
                               0511  1307
                               0511  1308          ; Test for user or far end device error (SS$_DEVREQERR)
                               0511  1309
          50    0334 8F   3C      0511  1310          MOVZWL  #SS$_DEVREQERR,R0       ; Assume yes
    51    00801BD0 8F   D3      0516  1311          BITL    #XF$M_IOS_DDIDIS!-      ; Test for DDI disable,
                               051D  1312            XF$M_IOS_RNGERR!-       ; range error,
                               051D  1313            XF$M_IOS_UNQERR!-       ; unaligned queue error,
                               051D  1314            XF$M_IOS_INVPKT!-       ; invalid packet,
                               051D  1315            XF$M_IOS_FREQMT!-       ; free queue empty,
```

```
                     051D  1316                    XF$M_IOS_INVDDI!-       ; invalid DDI command,
                     051D  1317                    XF$M_IOS_LENERR!-       ; length error,
                     051D  1318                   <XF$M_IOS_DDIERR @ -     ; and DDI error.
                     051D  1319                    XF$V_IOS_DDISTS>,R1     ; (DDIERR bit must be shifted over)
                05   12  051D  1320          BNEQ    70$                   ; Have one of those
                     051F  1321                                            ; Note:  Branch to 80$ if
                     051F  1322                                            ; SS$_DEVREQERR errors should
                     051F  1323                                            ; not be error logged.
                     051F  1324
                     051F  1325          ;
                     051F  1326          ; None of the error bits are set so we have to treat it as a
                     051F  1327          ; controller error (SS$_CTRLERR).  Fall through to ...
                     051F  1328          ;
                     051F  1329
          50   0054 8F   3C  051F  1330 60$:     MOVZWL  #SS$_CTRLERR,R0       ; Status
                     0524  1331
 00000000'GF   16   0524  1332 70$:     JSB     G^ERL$DEVICERR                ; Log device error
                     052A  1333
                     052A  1334 80$:     ; Common return
                     052A  1335
                05   052A  1336          RSB
```

```
                     052B  1338          .SBTTL  QUEUE_PKT_AST - Queue packet AST
                     052B  1339
                     052B  1340 ;++
                     052B  1341 ; FUNCTIONAL DESCRIPTION:
                     052B  1342 ;
                     052B  1343 ;       This routine is called to queue an AST to the user process and
                     052B  1344 ;       set an event flag to indicate that a packet was placed on the
                     052B  1345 ;       process's termination queue.  Both queueing the AST and setting
                     052B  1346 ;       the event flag are optional; either or both may occur.  An AST
                     052B  1347 ;       is queued if a packet AST address is specified in the I/O packet
                     052B  1348 ;       and the event flag is set if the modifier IO$_SETEVF is present
                     052B  1349 ;       in the I/O subfunction code.  Note that this routine forks to
                     052B  1350 ;       Queue AST IPL before queueing the AST or setting the event flag.
                     052B  1351 ;
                     052B  1352 ; CALLING SEQUENCE:
                     052B  1353 ;
                     052B  1354 ;       BSBW    QUEUE_PKT_AST
                     052B  1355 ;
                     052B  1356 ; INPUT PARAMETERS:
                     052B  1357 ;
                     052B  1358 ;       R3      Address of I/O packet
                     052B  1359 ;       R5      Address of UCB
                     052B  1360 ;
                     052B  1361 ; IMPLICIT INPUTS:
                     052B  1362 ;
                     052B  1363 ;       Various fields in the I/O packet
                     052B  1364 ;
                     052B  1365 ; OUTPUT PARAMETERS:
                     052B  1366 ;
                     052B  1367 ;       R0      Completion code
                     052B  1368 ;
                     052B  1369 ; IMPLICIT OUTPUTS:
                     052B  1370 ;
                     052B  1371 ;       None
                     052B  1372 ;
                     052B  1373 ; COMPLETION CODES:
                     052B  1374 ;
                     052B  1375 ;       SS$_NORMAL      Normal successful completion
                     052B  1376 ;       SS$_INSFMEM     Insufficient dynamic memory (to allocate an ACB)
                     052B  1377 ;       SS$_EXQUOTA     Exceeded AST quota
                     052B  1378 ;
                     052B  1379 ; SIDE EFFECTS:
                     052B  1380 ;
                     052B  1381 ;       R1,R2,R3,R4 and R5 are not preserved.  REPEAT R5!!!
                     052B  1382 ;--
                     052B  1383
                     052B  1384 QUEUE_PKT_AST:
                     052B  1385
                     052B  1386          ; Make sure the process has enough AST quota to allocate
                     052B  1387          ; a FORK/AST block.
                     052B  1388
  55   0C A3    3C   052B  1389          MOVZWL  IRP$L_PID(R3),R5        ; Get process index
00000000'GF    DD   052F  1390          PUSHL   G^SCH$GL_PCBVEC         ; Push address of PCB table
  55    9E45    D0   0535  1391          MOVL    @(SP)+[R5],R5           ; Get PCB address
  50      1C    3C   0539  1392          MOVZWL  #SS$_EXQUOTA,R0         ; Assume error
    38 A5       B5   053C  1393          TSTW    PCB$Q_ASTCNT(R5)        ; Enough AST quota left?
      1E        15   053F  1394          BLEQ    10$                     ; No!
```

```
                                                                J 1

                38 A5    B7  0541  1395          DECW    PCB$W_ASTCNT(R5)        ; Yes, take one away
                                 0544  1396
                                 0544  1397          ; Allocate a packet to be used as a fork block and AST control block
                                 0544  1398
          51    00C4 8F    3C  0544  1399          MOVZWL  #IRP$K_LENGTH,R1        ; Length = an I/O pkt because it's fast
                   53    DD  0549  1400          PUSHL   R3                      ; Save R3
           00000000'GF    16  054B  1401          JSB     G^EXE$ALONONPAGED       ; Returns pointer to packet in R2
                   53 8ED0  0551  1402          POPL    R3                      ; Restore R3
                09 50    E8  0554  1403          BLBS    R0,20$                  ; Successful allocation
          50    0124 8F    3C  0557  1404          MOVZWL  #SS$_INSFMEM,R0         ; Error - insufficient dynamic memory!
                38 A5    B6  055C  1405          INCW    PCB$W_ASTCNT(R5)        ; Add 1 back to AST quota
                         05  055F  1406  10$:     RSB                             ; Error return
                                 0560  1407
                                 0560  1408  20$:     ; Put size and type into packet and then fork to Queue AST IPL
                                 0560  1409          ; which returns success status to caller
                                 0560  1410
                                 0560  1411          ASSUME IRP$B_TYPE EQ IRP$W_SIZE+2
                                 0560  1412          ASSUME FKB$B_FIPL EQ IRP$B_RMOD
                                 0560  1413
    08 A2    000200C4 8F    D0  0560  1414          MOVL    #<DYN$C_ACB@16>+IRP$K_LENGTH,IRP$W_SIZE(R2)     ; Size and type
             08 A2    06    90  0568  1415          MOVB    #IPL$_QUEUEAST,FKB$B_FIPL(R2)   ; Set fork IPL = Queue AST IPL
                55    52    D0  056C  1416          MOVL    R2,R5                   ; R5 must point to fork block
                50    01    3C  056F  1417          MOVZWL  #SS$_NORMAL,R0          ; Return normal status to caller
                                 0572  1418          FORK                            ; Fork!
                                 0578  1419
                                 0578  1420          ; Build AST control block in preparation for queueing AST.
                                 0578  1421          ; R3 points to I/O packet, R5 points to AST control block.
                                 0578  1422
                                 0578  1423          ASSUME  IRP$L_PKTASTPRM EQ IRP$L_PKTASTADR+4
                                 0578  1424          ASSUME  ACB$L_ASTPRM EQ ACB$L_AST+4
                                 0578  1425
          51    0C A3    D0  0578  1426          MOVL    IRP$L_PID(R3),R1        ; Get PID and save for SCH$POSTEF
          0C A5    51    D0  057C  1427          MOVL    R1,ACB$L_PID(R5)        ; Store PID in ACB
                40 A3    7D  0580  1428          MOVQ    IRP$L_PKTASTADR(R3),-   ; Store packet AST address and parameter
                10 A5        0583  1429                  ACB$L_AST(R5)           ; in ACB
                0B A3    90  0585  1430          MOVB    IRP$B_RMOD(R3),-        ; Store access mode
                0B A5        0588  1431                  ACB$B_RMOD(R5)
    0B A5       40 8F    88  058A  1432          BISB    #ACB$M_QUOTA,ACB$B_RMOD(R5)  ; Set AST quota accounting flag
                                 058F  1433
                                 058F  1434  30$:     ; Now post event flag if subfunction code specifies it
                                 058F  1435
             52    01    9A  058F  1436          MOVZBL  #PRI$_IOCOM,R2          ; Priority incr. class = I/O complete
    0A 20 A3       06    E1  0592  1437          BBC     #IO$V_SETEVF,IRP$W_FUNC(R3),40$ ; Br. if don't post event flag
          53    22 A3    9A  0597  1438          MOVZBL  IRP$B_EFN(R3),R3        ; Get event flag number
       00000000'GF    16  059B  1439          JSB     G^SCH$POSTEF            ; Post event flag
                                 05A1  1440
                                 05A1  1441  40$:     ; Now either queue AST or deallocate AST control block.
                                 05A1  1442
                10 A5    D5  05A1  1443          TSTL    ACB$L_AST(R5)           ; Is AST specified?  (Address non-zero)?
                   06    13  05A4  1444          BEQL    50$                     ; No, deallocate ACB
          00000000'GF    17  05A6  1445          JMP     G^SCH$QAST              ; Yes, queue AST.  SCH$QAST returns to
                                 05AC  1446                                          ; caller.
                                 05AC  1447
                                 05AC  1448  50$:     ; Don't give AST, so deallocate packet.
                                 05AC  1449
          52    0C A5    3C  05AC  1450          MOVZWL  ACB$L_PID(R5),R2        ; But first increment AST quota
          00000000'GF    DD  05B0  1451          PUSHL   G^SCH$GL_PCBVEC         ; Push address of PCB table
```

```
                52   9E42   D0  05B6  1452          MOVL    @(SP)+[R2],R2           ; Get PCB address
                  38 A2    B6  05BA  1453          INCW    PCB$W_ASTCNT(R2)        ; Increment AST quota
                50   55    D0  05BD  1454          MOVL    R5,R0                   ; Address of packet
00000000'GF     17         05C0  1455              JMP     G^EXE$DEANONPAGED       ; Deallocate packet.
                           05C6  1456                                              ; returns to caller.
```

```
                                        .SBTTL  CANCEL_IO - Cancel I/O routine

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine performs the Cancel I/O function.  If the UCB is
                                ;       busy, then the channel index and process id in the IRP are
                                ;       compared with those passed by the caller.  If they match, then
                                ;       this routine calls ABORT which aborts the DR32.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSBW    CANCEL_IO
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       R2      Channel Index
                                ;       R3      Address of IRP
                                ;       R4      Current process PCB address
                                ;       R5      address of UCB
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       None
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       None
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       None
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       The I/O (if any) is completed with status SS$_ABORT
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       None
                                ;--
                                CANCEL_IO:
24 64 A5   08   E1              BBC     #UCB$V_BSY,UCB$W_STS(R5),60$  ; Branch if UCB is not busy
   28 A3   52   B1              CMPW    R2,IRP$W_CHAN(R3)             ; Compare channels
           1E   12              BNEQ    60$                           ; No match
0C A3   60 A4   D1              CMPL    PCB$L_PID(R4),IRP$L_PID(R3)   ; Compare process id's
           17   12              BNEQ    60$                           ; No match
                                ;
                                ; Channel and process id's match.  Check to make sure adapter
                                ; has power.  If it does, abort transfer.
                                ;
                                ASSUME  UCB_V_ADPPWRUP  EQ  0
                                ASSUME  IDB$L_CSR  EQ  0
                                ;
    13 68 A5   E9               BLBC    UCB$W_DEVSTS(R5),60$    ; Branch if adapter has no power
           54   DD               PUSHL   R4                      ; Save R4
```

```
                54   24 A5    D0  05DE  1515          MOVL    UCB$L_CRB(R5),R4            ; Get pointer to CRB
                54   2C B4    D0  05E2  1516          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get address of 1st device CSR
                   50    2C    3C  05E6  1517          MOVZWL  #SS$_ABORT,R0               ; Status
                     FE5F    30  05E9  1518          BSBW    ABORT                       ; Abort device
                         54 8ED0  05EC  1519          POPL    R4                          ; Restore R4
                                  05EF  1520
                         05      05EF  1521 60$:      RSB
```

```
                                        .SBTTL  REGDUMP - Register Dump Routine

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine copies relevant DR32 registers into either a diagnostic
                                ;       buffer or an error log buffer.  It is called from the error logging
                                ;       routine and from the diagnostic buffer fill routine.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB     REGDUMP
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       R0      Address of buffer to store registers
                                ;       R4      Address of first device CSR
                                ;       R5      Address of UCB
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       None
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       None
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       None
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       None
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       None
                                ;--

                                REGDUMP:
        03   E0  05F0            BBS     #UCB_V_DR750,-
    05 68 A5        05F2            UCB$W_DEVSTS(R5),10$    ; If set, yes DR750
                    05F5
     52   0F   D0  05F5            MOVL    #15,R2                  ; Number of reg. in first DR780 group
          03   11  05F8            BRB     20$                     ; Go start
                    05FA
     52   07   D0  05FA    10$:    MOVL    #7,R2                   ; Number of reg. in first DR750 group
                    05FD
     80   28   D0  05FD    20$:    MOVL    #40,(R0)+               ; Store number of registers to be saved
80   00A8 C5   D0  0600            MOVL    UCB$L_SAVDCR(R5),(R0)+  ; Store copy of saved DCR
                    0605
  51   04 A4   DE  0605            MOVAL   DR_UTL(R4),R1           ; Address of first register group
     80   81   D0  0609    30$:    MOVL    (R1)+,(R0)+             ; Store next register
       FA 52   F5  060C            SOBGTR  R2,30$                  ; Repeat
                    060F
          03   E0  060F            BBS     #UCB_V_DR750,-
```

```
                10 68 A5        0611  1580                      UCB$W_DEVSTS(R5),50$    ; If set, then DR750
                                0614  1581
                52    18  D0    0614  1582                MOVL  #24,R2                  ; Number of registers in second group
              51  0400  C4  DE  0617  1583                MOVAL DR_780_DSL(R4),R1       ; Address of second register group
                80    81  D0    061C  1584 40$:           MOVL  (R1)+,(R0)+             ; Store next register
                  FA  52  F5    061F  1585                SOBGTR R2,40$                 ; Repeat
                      10  11    0622  1586                BRB   70$
                                0624  1587
                52    20  D0    0624  1588 50$:           MOVL  #32,R2                  ; Number of registers in second group
                  08  A4  D4    0627  1589                CLRL  DR_WCSA(R4)             ; Set indirect register address to zero
              51  0C  A4  DE    062A  1590                MOVAL DR_WCSD(R4),R1          ; Address of data register
                80    61  D0    062E  1591 60$:           MOVL  (R1),(R0)+              ; Stores next register and bumps adderss
                  FA  52  F5    0631  1592                SOBGTR R2,60$                 ; Repeat
                                0634  1593
                      05  05    0634  1594 70$:           RSB
```

```
                        0635  1596          .SBTTL  UNIT_INIT - Unit initialization
                        0635  1597
                        0635  1598 ;++
                        0635  1599 ; FUNCTIONAL DESCRIPTION:
                        0635  1600 ;
                        0635  1601 ;       This routine is entered when the driver is loaded and on
                        0635  1602 ;       system power recovery.  On driver load, it initializes
                        0635  1603 ;       the UCB and sets the protection on the page containing
                        0635  1604 ;       the GO bit to user writeable.  It also determines which cpu type
                        0635  1605 ;       it is running on (11/780 or 11/750) and sets a bit in
                        0635  1606 ;       UCB$W_DEVSTS to indicate which one.
                        0635  1607 ;       On power recovery, it simply returns.  Power recovery is actually
                        0635  1608 ;       handled in the interrupt handler.
                        0635  1609 ;
                        0635  1610 ; CALLING SEQUENCE:
                        0635  1611 ;
                        0635  1612 ;       JSB     UNIT_INIT
                        0635  1613 ;
                        0635  1614 ; INPUT PARAMETERS:
                        0635  1615 ;
                        0635  1616 ;       R5      Address of UCB
                        0635  1617 ;
                        0635  1618 ; IMPLICIT INPUTS:
                        0635  1619 ;
                        0635  1620 ;       None
                        0635  1621 ;
                        0635  1622 ; OUTPUT PARAMETERS:
                        0635  1623 ;
                        0635  1624 ;       None
                        0635  1625 ;
                        0635  1626 ; IMPLICIT OUTPUTS:
                        0635  1627 ;
                        0635  1628 ;       None
                        0635  1629 ;
                        0635  1630 ; COMPLETION CODES:
                        0635  1631 ;
                        0635  1632 ;       None
                        0635  1633 ;
                        0635  1634 ; SIDE EFFECTS:
                        0635  1635 ;
                        0635  1636 ;       R0, R1, R2, AND R4 ARE NOT PRESERVED
                        0635  1637 ;--
                        0635  1638
                        0635  1639 UNIT_INIT:
                        0635  1640          ; Determine if this is initial loading or a power recovery.
                        0635  1641
5D 64 A5   05   E0      0635  1642          BBS     #UCB$V_POWER,UCB$W_STS(R5),INIT_DONE  ; Branch if power recovery
                        063A  1643
                        063A  1644          ; Get address of IDB and first device CSR
                        063A  1645
   51   24 A5   D0      063A  1646          MOVL    UCB$L_CRB(R5),R1          ; Get address of CRB
   52   2C A1   D0      063E  1647          MOVL    CRB$L_INTD+VEC$L_IDB(R1),R2  ; Get address of IDB
        54   62   D0      0642  1648          MOVL    IDB$L_CSR(R2),R4          ; Get address of first device CSR
                        0645  1649
                        0645  1650          ; Make UCB owner of IDB and reset DR.
                        0645  1651
   04 A2   55   D0      0645  1652          MOVL    R5,IDB$L_OWNER(R2)        ; Make UCB owner of IDB
```

```
                64   4000 8F   3C   0649  1653          MOVZWL  #DCR_K_RESET,DR_DCR(R4) ; Reset DR.
                64   0600 8F   3C   064E  1654          MOVZWL  #DCR_K_SETINTENB,DR_DCR(R4)     ; Enable interrupts
                                    0653  1655
                                    0653  1656          ; Set protection on page containing GO bit to user mode writeable.
                                    0653  1657
                50   0200 C4   DE   0653  1658          MOVAL   DR_USER(R4),R0              ; Get address of GO bit
           51   50     15   09   EF   0658  1659          EXTZV   #VA$V_VPN,#VA$S_VPN,R0,R1 ; Get virtual page number
            52   00000000'GF   D0   065D  1660          MOVL    G^MMG$GL_SPTBASE,R2         ; Get address of system page table
                   52   6241   DE   0664  1661          MOVAL   (R2)[R1],R2                 ; Get address of PTE that maps GO bit
                     1B   04   F0   0668  1662          INSV    #PRT$C_UW,#PTE$V_PROT,- ; Set protection in appropriate PTE
                     62   04        066B  1663                  #PTE$S_PROT,(R2)
                                    066D  1664          INVALID R0                          ; Invalidate translation buffer
                                    0670  1665
                                    0670  1666          ; Assume adapter has power so set adapter power up bit
                                    0670  1667
             68 A5   01   A8   0670  1668          BISW    #UCB_M_ADPPWRUP,UCB$W_DEVSTS(R5)
                                    0674  1669
                                    0674  1670          ; Now determine which type of DR32 we have by seeing
                                    0674  1671          ; what type of cpu we have.  Currently, the only DR32s
                                    0674  1672          ; supported are the DR780 and the DR750.  If we (somehow) get
                                    0674  1673          ; any other cpu type, about all we can do is not set the online
                                    0674  1674          ; bit in the UCB.  Also note that we set a bit in the UCB
                                    0674  1675          ; to indicate which type of DR32 we have.  In the rest of the driver
                                    0674  1676          ; we key off of this bit, rather than using the CPUDISP macro.
                                    0674  1677
                                    0674  1678          ASSUME  DT$_DR750  EQ  DT$_DR780+1
                                    0674  1679
             41 A5   02   90   0674  1680          MOVB    #DT$_DR780,UCB$B_DEVTYPE(R5)    ; Assume device type is DR780
                                    0678  1681
                                    0678  1682          CPUDISP <DR_780,DR_750,DR_730,DR_790>   ; * Dispatch on CPU type *
                                    068C  1683
          41 A5   96   068C  1684 DR_750: INCB    UCB$B_DEVTYPE(R5)               ; Make device type be DR750
68 A5   08   A8   068F  1685          BISW    #UCB_M_DR750,UCB$W_DEVSTS(R5)   ; Set DR750 bit in UCB
                                    0693  1686                                  ; Fall through to ...
                                    0693  1687
                                    0693  1688 DR_790:                          ; Same action as for DR780.
64 A5   10   A8   0693  1689 DR_780: BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; Set online bit
                                    0697  1690
                                    0697  1691 DR_730:
                                    0697  1692
                                    0697  1693 DR_END:                           ; * End of cpu dependent code *
                                    0697  1694
                                    0697  1695 INIT_DONE:
                05   0697  1696          RSB
                                    0698  1697
                                    0698  1698
                                    0698  1699
                                    0698  1700 XF_END:                                  ; End of driver
                                    0698  1701
                                    0698  1702
                                    0698  1703          .END
```

```
$$$                             = 00000020 R     02
$$OP                            = 00000002
ABORT                             0000044B R     03
ABORT_INT                         000004B2 R     03
ABORT_IO                          00000196 R     03
ACB$B_RMOD                      = 0000000B
ACB$L_AST                       = 00000010
ACB$L_ASTPRM                    = 00000014
ACB$L_PID                       = 0000000C
ACB$M_QUOTA                     = 00000040
ACCESS_VIO                        00000193 R     03
ALL_OK                            000001A5 R     03
AT$_DR                          = 00000002
BUG$_UNSUPRTCPU                   ******** X     03
CANCEL_IO                         000005C6 R     03
CRB$L_INTD                      = 00000024
DC$_REALTIME                    = 00000060
DCR_K_CLRABTINT                 = 00000400
DCR_K_CLRHLT                    = 00000700
DCR_K_CLRPKTINT                 = 00003000
DCR_K_CLRPWRDN                  = 00000200
DCR_K_CLRPWRUP                  = 00000100
DCR_K_RESET                     = 00004000
DCR_K_SETEXTABT                 = 00002000
DCR_K_SETINTENB                 = 00000600
DDB$L_DDT                       = 0000000C
DEV$M_AVL                         ******** X     02
DEV$M_ELG                         ******** X     02
DEV$M_IDV                         ******** X     02
DEV$M_ODV                         ******** X     02
DEV$M_RTM                         ******** X     02
DONE                              0000046E R     03
DPT$C_LENGTH                    = 00000038
DPT$C_VERSION                   = 00000004
DPT$INITAB                        00000038 R     02
DPT$REINITAB                      0000004B R     02
DPT$TAB                           00000000 R     02
DR750_MAXRATE                   = 000000FC
DR780_MAXRATE                   = 000000FB
DR_730                            00000697 R     03
DR_750                            0000068C R     03
DR_750_BFRBVA                     00000006
DR_750_BFRLEN                     00000007
DR_750_BFRSVAPT                   00000008
DR_750_CMDBVA                     00000003
DR_750_CMDLEN                     00000004
DR_750_CMDSVAPT                   00000005
DR_750_DSL                        00000000
DR_750_GBR                        00000002
DR_750_SBR                        00000001
DR_780                            00000693 R     03
DR_780_BFRBVA                     00000418
DR_780_BFRLEN                     0000041C
DR_780_BFRSVAPT                   00000420
DR_780_CMDBVA                     0000040C
DR_780_CMDLEN                     00000410
DR_780_CMDSVAPT                   00000414
DR_780_DSL                        00000400
DR_780_GBR                        00000408
DR_780_SBR                        00000404
DR_790                            00000693 R     03
DR_DCR                            00000000
DR_DCR_M_DCRABT                 = 00040000
DR_DCR_M_DCRHLT                 = 00020000
DR_DCR_M_ENPEAB                 = 08000000
DR_DCR_M_ID1ERR                 = 00001000
DR_DCR_M_ID1TO                  = 00004000
DR_DCR_M_ID2ERR                 = 00000100
DR_DCR_M_ID2TO                  = 00000400
DR_DCR_M_PARERR                 = 80000000
DR_DCR_M_PKTINT                 = 00080000
DR_DCR_M_PWR_DN                 = 00800000
DR_DCR_M_PWR_UP                 = 00400000
DR_DCR_V_DCRABT                 = 00000012
DR_DCR_V_DCRHLT                 = 00000011
DR_DCR_V_EXTABT                 = 00000018
DR_DCR_V_PKTINT                 = 00000013
DR_DCR_V_PWR_DN                 = 00000017
DR_DCR_V_PWR_UP                 = 00000016
DR_DCR_V_RDS                    = 0000000F
DR_DCR_V_WCSPE                  = 0000001C
DR_DDIBCNT                        0000001C
DR_END                            00000697 R     03
DR_SBIADR                         00000014
DR_SBIBCNT                        00000018
DR_USER                           00000200
DR_UTL                            00000004
DR_UTL_M_ENPEAB                 = 08000000
DR_UTL_M_PARERR                 = 80000000
DR_UTL_M_VALID                  = 00000800
DR_UTL_V_PARERR                 = 0000001F
DR_UTL_V_WCSPE                  = 0000001C
DR_WCSA                           00000008
DR_WCSA_M_SEL                   = 00000001
DR_WCSA_M_WCS                   = 80000000
DR_WCSA_V_ADDR                  = 00000001
DR_WCSA_V_SEL                   = 00000000
DR_WCSD                           000G000C
DT$_DR750                       = 00000003
DT$_DR780                       = 00000002
DYN$C_ACB                       = 00000002
DYN$C_CRB                       = 00000005
DYN$C_DDB                       = 00000006
DYN$C_DPT                       = 0000001E
DYN$C_IRPE                      = 0000002C
DYN$C_UCB                       = 00000010
EMB$L_DV_REGSAV                 = 0000004E
ERL$DEVICERR                      ******** X     03
ERL$DEVICTMO                      ******** X     03
EXE$ABORTIO                       ******** X     03
EXE$ALLOCIRP                      ******** X     03
EXE$ALONONPAGED                   ******** X     03
EXE$DEANONPAGED                   ******** X     03
EXE$FINISHIOC                     ******** X     03
```

```
EXE$FORK                    ********  X   03
EXE$GB_CPUTYPE              ********  X   03
EXE$IOFORK                  ********  X   03
EXE$MODIFYLOCKR             ********  X   03
EXE$QIODRVPKT               ********  X   03
EXE$WRITELOCK               ********  X   03
FINISH_IO                   0000019C  R   03
FKB$B_FIPL                = 0000000B
FUNCTABLE                   00000038  R   03
FUNCTAB_LEN               = 00000028
HANDLE_INT                  000003D4  R   03
IDB$L_CSR                 = 00000000
IDB$L_OWNER               = 00000004
INIT_DONE                   00000697  R   03
INTERRUPT_SVC               00000366  R   03
IO$V_SETEOF               = 00000006
IO$_LOADMCODE             = 00000001
IO$_STARTDATA             = 00000038
IO$_STARTDATAP            = 00000006
IO$_VIRTUAL               = 0000003F
IOC$DIAGBUFILL              ********  X   03
IOC$GW_XFMXRATE             ********  X   03
IOC$MNTVER                  ********  X   03
IOC$REQCOM                  ********  X   03
IOC$RETURN                  ********  X   03
IOC$WFIKPCH                 ********  X   03
IPL$_QUEUEAST             = 00000006
IRP$B_EFN                 = 00000022
IRP$B_FLAGS               = 0000003D
IRP$B_RATE                = 0000003C
IRP$B_RMOD                = 0000000B
IRP$B_TYPE                = 0000000A
IRP$K_LENGTH              = 000000C4
IRP$L_ABCNT               = 00000040
IRP$L_EXTEND              = 00000054
IRP$L_IOST2               = 0000003C
IRP$L_OBCNT               = 00000044
IRP$L_PID                 = 0000000C
IRP$L_PKTASTADR           = 00000040
IRP$L_PKTASTPRM           = 00000044
IRP$L_SVAPTE              = 0000002C
IRP$M_EXTEND              = 00000800
IRP$W_CHAN                = 00000028
IRP$W_FUNC                = 00000020
IRP$W_SIZE                = 00000008
IRP$W_STS                 = 0000002A
IRPE$B_TYPE               = 0000000A
IRPE$L_BBLKADR            = 00000048
IRPE$L_BCNT1              = 00000034
IRPE$L_BCNT2              = 00000040
IRPE$L_CBLKADR            = 00000044
IRPE$L_SVAPTE1            = 0000002C
IRPE$L_SVAPTE2            = 00000038
IRPE$W_BOFF1              = 00000030
IRPE$W_BOFF2              = 0000003C
IRPE$W_STS                = 0000002A
LOAD_MICROCODE              00000060  R   03
LOCK_BFR                    00000215  R   03
MASKA                     = 01000000
MASKL                     = 00000040
MMG$GL_GPTBASE              ********  X   03
MMG$GL_SPTBASE              ********  X   03
MMG$UNLOCK                  ********  X   03
P1                        = 00000000
P2                        = 00000004
P3                        = 00000008
PCB$L_PID                 = 00000060
PCB$W_ASTCNT              = 00000038
PR$_IPL                   = 00000012
PR$_SBR                   = 0000000C
PR$_SID_TYP780            = 00000001
PR$_TBIS                  = 0000003A
PRI$_IOCOM                = 00000001
PRT$C_UW                  = 00000004
PTE$S_PROT                = 00000004
PTE$V_PROT                = 0000001B
QUEUE_PKT_AST               0000052B  R   03
REGDUMP                     000005F0  R   03
REQ_COMPLETE                0000049C  R   03
SCH$GL_PCBVEC               ********  X   03
SCH$POSTEF                  ********  X   03
SCH$QAST                    ********  X   03
SIZ...                    = 00000001
SS$_ABORT                 = 0000002C
SS$_ACCVIO                = 0000000C
SS$_BADPARAM              = 00000014
SS$_BUFNOTALIGN           = 00000324
SS$_CTRLERR               = 00000054
SS$_DEVACTIVE             = 000002C4
SS$_DEVREQERR             = 00000334
SS$_EXQUOTA               = 0000001C
SS$_INSFMEM               = 00000124
SS$_IVBUFLEN              = 0000034C
SS$_MCNOTVALID            = 0000035C
SS$_NORMAL                = 00000001
SS$_PARITY                = 000001F4
SS$_POWERFAIL             = 00000364
SS$_TIMEOUT               = 0000022C
STARTDATA_FDT               00000146  R   03
STARTIO                     00000254  R   03
TIMEOUT                     0000046F  R   03
UCB$B_DEVCLASS            = 00000040
UCB$B_DEVTYPE             = 00000041
UCB$B_DIPL                = 0000005E
UCB$B_FIPL                = 0000000B
UCB$K_SIZE                = 000000AC
UCB$L_CRB                 = 00000024
UCB$L_DCR                   000000A0
UCB$L_DEVCHAR             = 00000038
UCB$L_DEVDEPEND           = 00000044
UCB$L_DPC                 = 0000009C
UCB$L_IRP                 = 00000058
UCB$L_SAVDCR                000000A8
UCB$L_SAVSTATUS             000000A4
```

```
UCB$M_ONLINE                    = 00000010
UCB$M_POWER                     = 00000020
UCB$V_BSY                       = 00000008
UCB$V_POWER                     = 00000005
UCB$W_DEVSTS                    = 00000068
UCB$W_STS                       = 00000064
UCB_M_ABORT                     = 00000004
UCB_M_ADPPWRUP                  = 00000001
UCB_M_DR750                     = 00000008
UCB_M_FKLOCK                    = 00000002
UCB_V_ABORT                     = 00000002
UCB_V_ADPPWRUP                  = 00000000
UCB_V_DR750                     = 00000003
UCB_V_FKLOCK                    = 00000001
UNIT_INIT                         00000635 R      03
VA$S_BYTE                       = 00000009
VA$S_VPN                        = 00000015
VA$V_VPN                        = 00000009
VEC$L_IDB                       = 00000008
VEC$L_UNITINIT                  = 00000018
WAIT                              00000450 R      03
XF$DDT                            00000000 RG     03
XF$K_CMT_LENGTH                 = 00000020
XF$L_CMT_BBLKSZ                 = 00000008
XF$L_CMT_CBLKAD                 = 00000004
XF$L_CMT_CBLKSZ                 = 00000000
XF$L_CMT_GBITAD                 = 0000001C
XF$M_IOS_BUSERR                 = 08000000
XF$M_IOS_CIPE                   = 40000000
XF$M_IOS_DDIDIS                 = 00000010
XF$M_IOS_DDIERR                 = 00000080
XF$M_IOS_DIPE                   = 80000000
XF$M_IOS_FREQMT                 = 00000200
XF$M_IOS_INVDDI                 = 00000800
XF$M_IOS_INVPKT                 = 00000100
XF$M_IOS_INVPTE                 = 00000004
XF$M_IOS_LENERR                 = 00001000
XF$M_IOS_RDSERR                 = 10000000
XF$M_IOS_RNGERR                 = 00000040
XF$M_IOS_UNQERR                 = 00000080
XF$M_IOS_WCSPE                  = 20000000
XF$V_CMT_DIPEAB                 = 00000001
XF$V_CMT_SETRTE                 = 00000000
XF$V_IOS_BUSERR                 = 0000001B
XF$V_IOS_DDISTS                 = 00000010
XF$V_IOS_RDSERR                 = 0000001C
XF$V_IOS_WCSPE                  = 0000001D
XF_END                            00000698 R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000424 (1060.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	0000005B (91.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00000698 (1688.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.02	00:00:01.94
Command processing	105	00:00:00.34	00:00:05.57
Pass 1	627	00:00:19.10	00:01:08.84
Symbol table sort	0	00:00:02.83	00:00:09.04
Pass 2	290	00:00:04.26	00:00:16.90
Symbol table output	34	00:00:00.17	00:00:00.62
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1093	00:00:26.75	00:01:42.93

The working set limit was 2400 pages.
159143 bytes (311 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2605 non-local and 65 local symbols.
1703 source lines were read in Pass 1, producing 20 object records in Pass 2.
60 pages of virtual memory were used to define 56 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	40
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	51

2870 GETS were required to define 51 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:XFDRIVER/OBJ=OBJ$:XFDRIVER MSRC$:XFDRIVER/UPDATE=(ENH$:XFDRIVER)+EXECML$/LIB

XFDRIVER
LIS

XGDRIVER
LIS

XIDRIVER
LIS